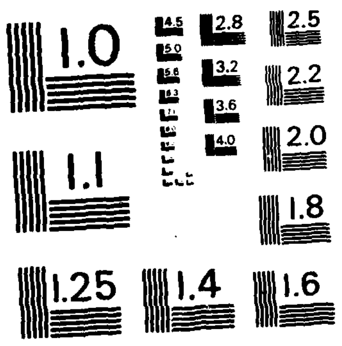
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

1

AD________

SLEEP-WAKEFULNESS DETERMINATIONS FROM HEART RATE DATA

Final Report

P. C. Richardson

August 1978

Supported by

US Army Medical Research and Development Command
Fort Detrick, Frederick, Maryland 21701

Contract No. DAMD17-74-C-4081

Electronics Research Center
The University of Texas at Austin
Austin, Texas 78712

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. A139436	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SLEEP-WAKEFULNESS DETERMINATIONS FROM HEART RATE DATA		5. TYPE OF REPORT & PERIOD COVERED Final 1 May 74 - 31 Aug 78
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) P.C. Richardson, A.J. Welch, H. Louise Eagle, and Martin Nitzberg		8. CONTRACT OR GRANT NUMBER(s) DAMD17-74-C-4081
9. PERFORMING ORGANIZATION NAME AND ADDRESS Electronics Research Center The University of Texas at Austin Austin, Texas 78712		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62771A.3E162771A804.00.005
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Medical Research and Development Command Fort Detrick, Frederick, Maryland 21701		12. REPORT DATE Aug 1978
		13. NUMBER OF PAGES 70
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

INTRODUCTION

Sleep has been studied from a very large number of points of view; from studies involving sleep deprivation, examination of sleep hemodynamics, studies examining penile erection during sleep, schizophrenia, gastric motility and circadian rhythms(51) and even biochemical and seasonal influences on sleep. The vastness of research conducted on sleep can probably best be perceived after reviewing some of the 4,300 references in the bibliography prepared by Nathaniel Kleitman(35) entitled, "Sleep and Wakefulness".

Our own investigation into the relationship between sleep depth and instantaneous beat-by-beat heart rate has been carried out intermittently over the past ten years, (~~1,2,39,~~58-62). Many properties of the relationship between heart beat patterns and the quantity and quality of sleep have been investigated in our laboratory.

The technical advances in small computers has created an upsurge in research to find an automated method of determining sleep quantity and quality. While the hardware capabilities of microprocessors have reached the stage of feasibility for automated sleep staging, software development has not kept pace. Thus, an automated on-line sleep evaluation device still remains a desirable yet unattained reality.

For the past decade, a number of projects have been conducted in the Biomedical Engineering Program here at the University of Texas regarding the development of automated sleep scoring software. Initially our studies were prompted by the advent of the space age where it became important to devise a method of determining levels of alertness of astronauts during prolonged space flight. Specifically, these projects involved bandwidth reduction of sleep, extraction of sleep information from heart rate data, analysis of sleep cycles, detection of REM, 1 sleep stage of eye movement from beat-by-beat heart rate, classification of sleep into awake, REM-1, and Stage 2, 3 and 4 using the "beatquency" domain. All of these projects were undertaken with a common goal in mind. That is, the development of an automated process by which <u>rapid</u>, <u>inexpensive</u> determinations of levels of alertness could be determined <u>accurately</u> using an <u>easily obtained</u> physiologic parameter. The physiologic parameter chosen in our studies was the beat-by-beat heart rate. Our investigations have produced some insight into this problem, but we have not realized the goal.

BACKGROUND

We do not really know what sleep is(17,26); however, we all sleep approximately one-third of our lives away. We know that if we do not sleep, we function poorly, and if the quantity and quality of our sleep is significantly altered we do not function with our usual capabilities(27,31,55).

With the advent of the polygraph and sleep staging, a method of quantitation and gradation of sleep became possible. While it would seem true that some insight into the nature and function of sleep might result from this quantification, sleep staging has merely raised additional unanswered questions about sleep quality.

Electroencephalographic (EEG) patterns obtained from polygraph recordings exhibit definitive characteristics in both awake and sleeping stages. Descriptions of these patterns employ objective terms such as alpha, beta, delta and theta waves, low voltage fast electroencephalographic waves, high voltage slow waves, spindle activity, K-complexes and the like; however, EEG, like sleep itself, exists as a continuous process rather than as a sequence of four discrete states.

With the discovery of ocular motility during sleep(4,5,19,46), another measure became available for objective observation. The electro-oculogram (EOG) measures electrical activity caused by eye movements. The correlation of concurrent stage 1 EEG sleep and EOG patterns with dreaming led to the definition of REM (rapid eye movement) sleep.

The classical technique for determining sleep levels or sleep stage involves the clinical interpretation of concomitant changes in the EEG and EOG patterns of polygraph traces on a continuous all night strip chart record. With these findings, researchers have developed standard procedures for "scoring" EEG patterns into different depths of sleep. This scoring process is a standard, somewhat tedious and time consuming process. Nevertheless, this method has provided us with quantitative techniques for research and has allowed us to perform extensive analyses of sleep and its related phenomena.

While each sleep researcher tends to develop his own method of quantitating sleep depth, sleep is typically divided into four primary stages: Stages awake (or zero), I, II, III and IV, with Stage IV representing the deepest sleep(16). Stage REM sleep (labeled Stage V) produces the same EEG patterns as Stage I, but during REM sleep there is intermittent rapid conjugate movement of the eyes. Typical patterns for EEG and EOG seen during these stages are shown in Figure 1(35). A graphical representation of staging throughout the entire night of sleep is shown in Figure 2(39). This figure depicts the sleep subject progressing from Stage awake or Stage IV and then returning rapidly to Stage I, REM, which is Stages I and REM combined. This pattern is typical of a "normal" night of sleep. The horizontal bars above Stage I, REM, indicate the occurrence of rapid conjugate eye movements, or Stage REM (V). It can also be seen that the subject makes a similar progression from light

SLEEP EEG PATTERNS

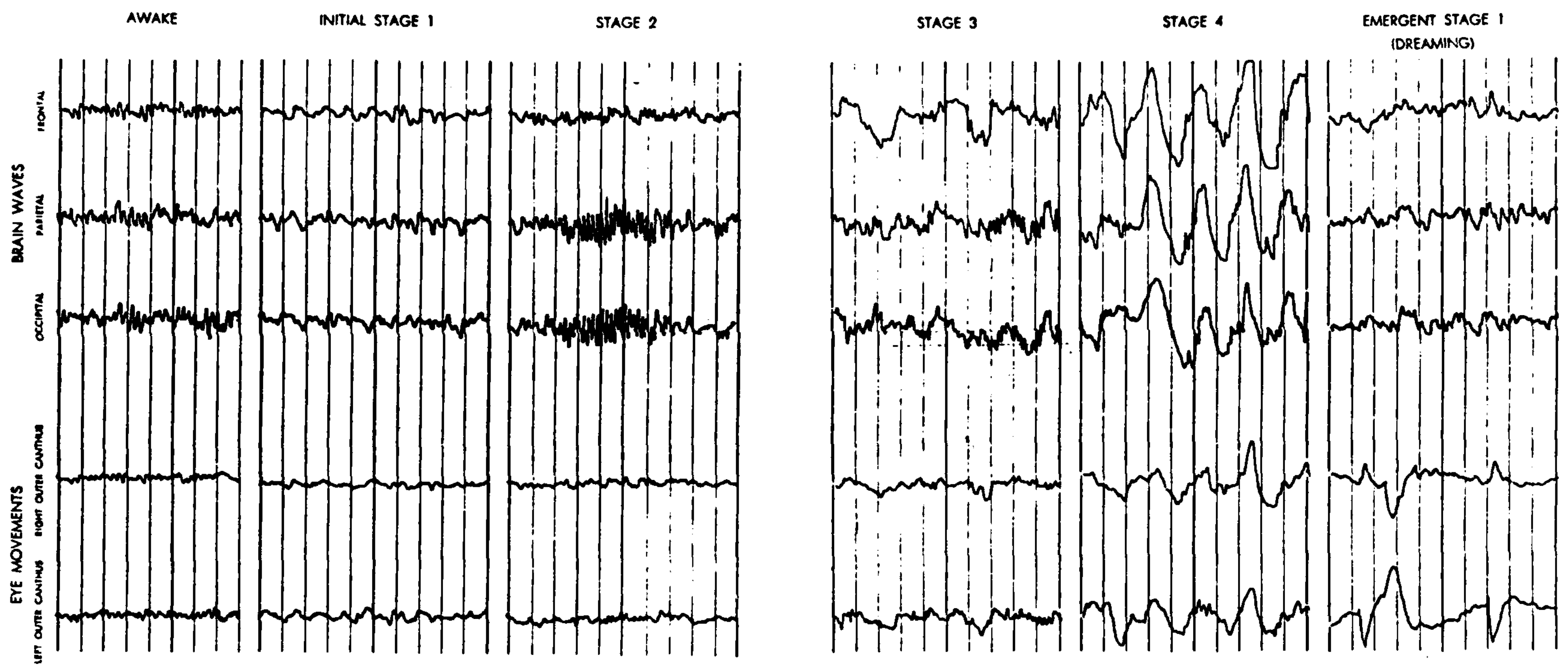

Figure 1.

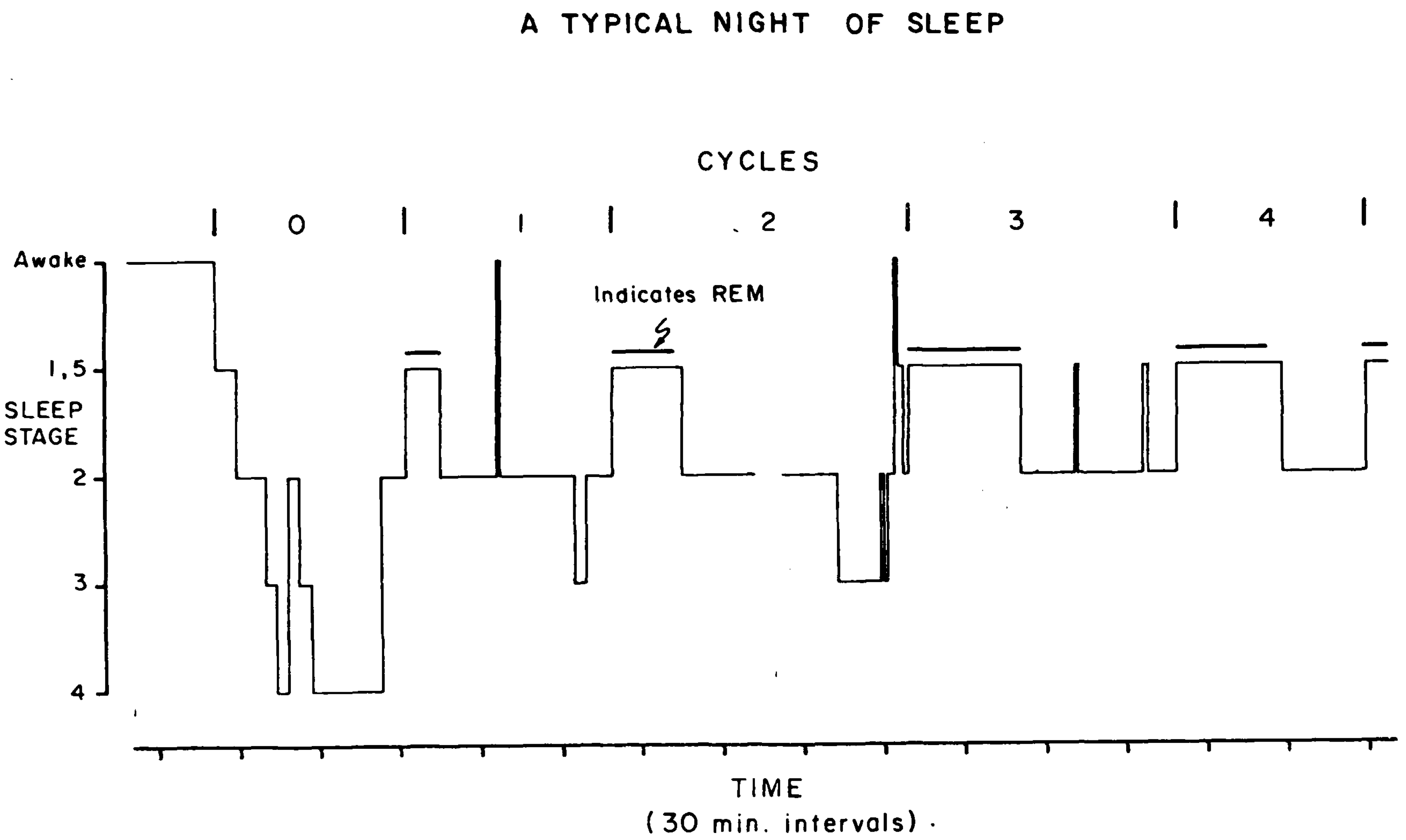
A TYPICAL NIGHT OF SLEEP
CYCLES
0
1
2
3
4
Awake
1,5
SLEEP STAGE
2
3
4
Indicates REM
TIME
(30 min. intervals)

Figure 2.

sleep to deep sleep throughout the night. This "cycling" occurs approximately every 60-100 minutes and has been described in numerous reports(1,9,11,16,34,44,57). We have defined the cycles as beginning at the onset of the REM period (horizontal bars) and ending at the onset of the next REM period, except for Cycle 0, which typically begins with Stage I where there are no rapid conjugate eye movements. The sleep subjects' record in produced five complete sleep cycles during this night of sleep. Usually, with each sleep cycle the "trough" of the cycle gets more shallow as the night progresses, which is also typical of the normal night of sleep(16,34). In general, progression from wakefulness to Stage IV at the beginning Cycle 0 is a continuum of change, while reversion to the lighter stages at the end of the cycle are quite abrupt(16). Stages III and IV are only occasionally reached during sleep cycles following the second. Another characteristic of sleep cycling is that the length of the REM periods increases as the night of sleep progresses, while non-REM durations decrease(4,14,25).

A considerable amount of research is currently in progress dedicated to a better understanding of the relationship of sleep patterns and heart rate patterns in infants who are apt to experience unexplained crib death(8,24,37,47,56). In addition, it is well known that sleep patterns of man differ from those of other mammals(30,49,53); however, sleep in man, while variable from night to night, is relatively consistent in terms of the pattern of sleep cycling within one subject and from one subject to another. Age plays an important role in the characteristics of sleep patterns(21).

It should be remembered that although sleep is described in quantitative measures or "stages," sleep patterns are associated with a continual process. This means that sleep levels do not always lend themselves to discrete classifications. In other words, there are fluctuations of depth within each stage as well as between stages(16,63,64).

The measurement of a variety of physiologic parameters associated with autonomic activity are known to be associated with changes in sleep patterns(3,12,15,17,27,32,35,43,52,53,54,63). In fact, the depth of sleep is said to take on different meanings according to the variables used to define it. The determination of depth should take into account the simultaneous fluctuations of heart rate, blood pressure, respiration and even skin potential. The measurement of gastric motility during sleep, metabolic rate, penile erections, peripheral muscle activity and pupillary reactions have also been regarded as autonomic correlates of sleep. Before the advent of the polygraph, autonomic variables were the only means of describing sleep(63). Summarizing as best is possible the commonality in these autonomic correlates of sleep depth research are:

1) In general, the trend is for heart rate, blood pressure and respiratory rate to decrease with increasing depth of sleep.

2) In general, heart rate, blood pressure and respiration increase their variability during REM sleep and decrease their variability with increasing depth of sleep.

3) Gross body movements exhibit cyclic variations closely associated with sleep cycles.

4) Skin potentials generally increase with the onset of deep sleep and are markedly reduced during REM sl^ep.

Concerning heart rate, in qeneral it is found at the heart slows from two to twelve beats per minute wit the onset of sleep(52). And on the average, the range of hea+ -ate is lower during sleep than during waking. Virtually all researchers agree(1,3,10,11,52,54,63) that the variability of heart rate is the most pronounced feature when considering different depths of sleep, since average heart rates overlap for most stages of sleep. Fluctuations in the heart rate are more erratic during light sleep as in Stages I, II, and REM and the fluctuations are slow and rhythmic in deeper sleep stages such as Stages III and IV. REM sleep is associated with the most outstanding and violent variations in heart rates; yet, paradoxically, it also contains periods of almost complete heart rate quiescence(10,52).

Both blood pressure and respiratory rate(11) follow a pattern and trend similar to heart rate during the sleep cycling process. The concensus view is that blood pressure and respiration show high degrees of variability during REM sleep and become more stable in deeper stages of sleep(3,15,25,33,52,63) On the average, blood pressure and respiratory rate during REM sleep are significantly higher. Blood pressure tends to reach its minimum during the first third of a night of sleep and is probably associated with the preponderance of Stages III and IV, which occur almost exclusively during this period of the night. Respiratory changes such as increased respiratory rate and decreased amplitude have been correlated with REM activity (periods when the eyes are actually moving)[3,6]. Body movements occur periodically throughout the night particularly associated with the deeper sleep at the end of each sleep cycle. Typically, body movements are seen to occur during Stage IV sleep just prior to REM[16,18]. The REM period is characterized by no peripheral activity, but movement begins again immediately after the REM and then returns toward quiescence with deepening sleep. The almost complete body paralysis during REM periods exists in sharp contrast to the violent autonomic activity associated with REM periods and is probably necessary to avoid destructive behavior and/or self-injury during REM sleep.

Changes in functions mediated by autonomic information such as heart rate, blood pressure and respiratory rate are thought to follow patterns mediated through parasympathetic and sympathetic neural control[7,33]. In some cases, these variables appear to be interdependent upon one another. For example, Bond et al.[10], found that heart rate variability during Stages I and II was rhythmic and roughly associated with respiration. During deeper sleep, as in Stages III and IV, there appeared to be an even intercorrelation and synchrony between heart rate patterns and respiratory activity. However, during REM periods there was total disassociation between heart rate and respiration. Snyder, et al.[52], also reported that heart rates, blood pressure and respiration "oscillated quite regularly around the relatively stable baseline" during Stages II and III/IV combined, while Stages I and REM combined featured wide and erratic fluctuations. Snyder reported many instances where changes in all three variables occurred together and in other instances completely independent change. They suggested that simultaneous changes of these variables are probably secondary to changes in respiration. Buast, et al.(7) performed detailed experiments on parasympathetic and sympathetic influences on heart rate during sleep using the cat. He concluded that phasic changes in the heart rate during sleep and wakefulness are decisively influenced by autonomic events. In general, he found that the fall in heart rate from wakefulness to synchronized (Stages II, III, IV in the human) sleep is due mainly to increased parasympathetic, or vagal, activity and that phasic changes during bursts of rapid eye movement are associated with phasic events in both the parasympathetic and sympathetic nerves connected to the heart. In other words, both types of autonomic activities play a role in physiologic changes during sleep.

There is a wide variety of automated methods of detecting and quantitating sleep. Larsen(38) suggests that motivation behind automated sleep stage research is to 1) provide a better understanding of the physiologic aspects of sleep and 2) to test the adequacy of variables thought to contain quantitative information about sleep.

Computerized sleep staging involves the seeking out of definitive measures of the EEG and then developing some analog, digital or hybrid method to use the measures for classification. In essence, researchers have attempted to devise automated methods of pattern recognition especially designed for sleep patterns of the EEG.

Approaches to the solution of the problem have employed techniques in spectral analysis(23,36,42,45), period analysis(51), and baseline cross-analysis(28,38,60). These procedures have met with only limited success except for the hybrid system developed by Gaillard(22), et al.,who reported overall accuracy (when compared with visual scoring) up to 92%. An overview of the results of several studies is shown in Table I. Problems of classifications are cited as overlap of measures of discrimination(51), the need of inclusion of more measures(28,42,51), imprecise definition of variables(28), unreliability of visual scoring(42), and the need for additional methods of analysis. All of the above methods of automated sleep staging require the processing of extensive amounts of data and considerable amounts of computer time(29).

Our research program is dedicated to reducing the data bulk and yet providing accurate, reliable, quantitative measures of sleep. Using heart rate as our input variable has certain advantages over previously cited measures. For example:

1) the ECG is an easily measured physiologic parameter,
2) practically speaking, the ECG alleviates the need of uncomfortable multiple electrodes placed in the scalp used in monitoring EEG, which are cumbersome and restrictive of movement, and

TABLE I

REPORTED RESULTS

OF AUTOMATED CLASSIFICATION

Stage	Roessler[51]	Lubin[42]	Larsen[38]	Itil[28]
W	87%	64%	91%	74%
1	45%	70%	64%	73%
REM*	-	56%	66%	-
2	63%	82%	85%	55%
3	79%	0%	85%	62%
4	84%	91%	85%	59%
OVERALL	69%	60%	79%	65%

*Most studies did not attempt to classify REM separately.

3) in order to completely describe beat-by-beat heart rate only 50-100 samples per <u>minute</u> are needed, whereas for digital description of EEG requires 250-500 samples per <u>second</u>!

Hence, based upon what is known about heart rate and sleep; and our desire for a low cost, easily derived input parameter of limited bulk, we have attempted to design an automated method for the classification of sleep patterns using beat-by-beat heart rate.

THE PROBLEM

The objective of this research is the development of a process by which rapid, inexpensive determination of level of sleep can be performed accurately using an easily derived physiologic parameter such as the beat-by-beat heart rate.

OUR PREVIOUS WORK

Our first study reported computer sleep stage classification using heart rate data and this was an overview of a two-year research effort in the bandwidth reduction of sleep information(61,62). Our second study investigated intrasubject variability that we observed during our first study(1,2). This study involved variations of heart rate during sleep as a function of sleep cycle. A third study provided another integral part of our research plan. It was dedicated to the detection of REM 1 sleep stage and eye movement from beat-by-beat heart rate(58). In this study we hypothesized that sudden cyclic changes in heart rate observed during sleep were caused by the same neural process which caused rapid eye movements. Our fourth study involved an attempt on our part to generalize the work of the previous years to other subjects and to check intersubject variability(40,41). Our classification technique used what we have since come to call the "beatquency domain". Our fifth study(41) utilized our previously developed algorithms on a wider data base to check for intrasubject variability with repeated measurements of sleep on the same subject over a 7-week period of time. In addition, work was performed using a supervised and an unsupervised pattern classification techniques[50]. Our final project of this study was dedicated to a better understanding of the difference between awake and sleep heart rate data on a new data base that was collected here at the University of Texas.

METHODS

AWAKE/SLEEP DATA BASE

Analog sleep pattern recordings were made on seven normal subjects. These analog tapes were recorded in our sleep laboratory at the University of Texas during August 1977 and contain a total of five channels per subject: one channel of ECG, two channels of EOG, and two channels of EEG. In addition, the 14 channel magnetic tapes contain a slow time code (hrs/min) and a fast time code (modulated 1 kHz sinusoidal). These data were recorded on continuous paper strip charts which we used for the visual scanning and scoring by an experienced sleep scorer. Each night was hand scored by an expert using the Manual of Standardized Terminology Technique and Scoring for Sleep Stages of Human Subjects by Rechtschaffen and Kales(48).

DIGITIZATION OF DATA

Three subjects' analog ECG records were selected for digitizing. For each subject, comparable amounts of awake and sleep data were digitized as follows:

Record 5 - 36 min of awake, 40 min of sleep

Record 6 - 68 min of awake, 66 min of sleep

Record 7 - 32 min of awake, 32 min of sleep

The interval between successive ECG R-waves (the R-to-R interval) was measured and recorded using a set-up schematically represented in Figure 3. The ECG analog records were played using a Sangamo 14 channel FM tape recorder, amplified to 2½

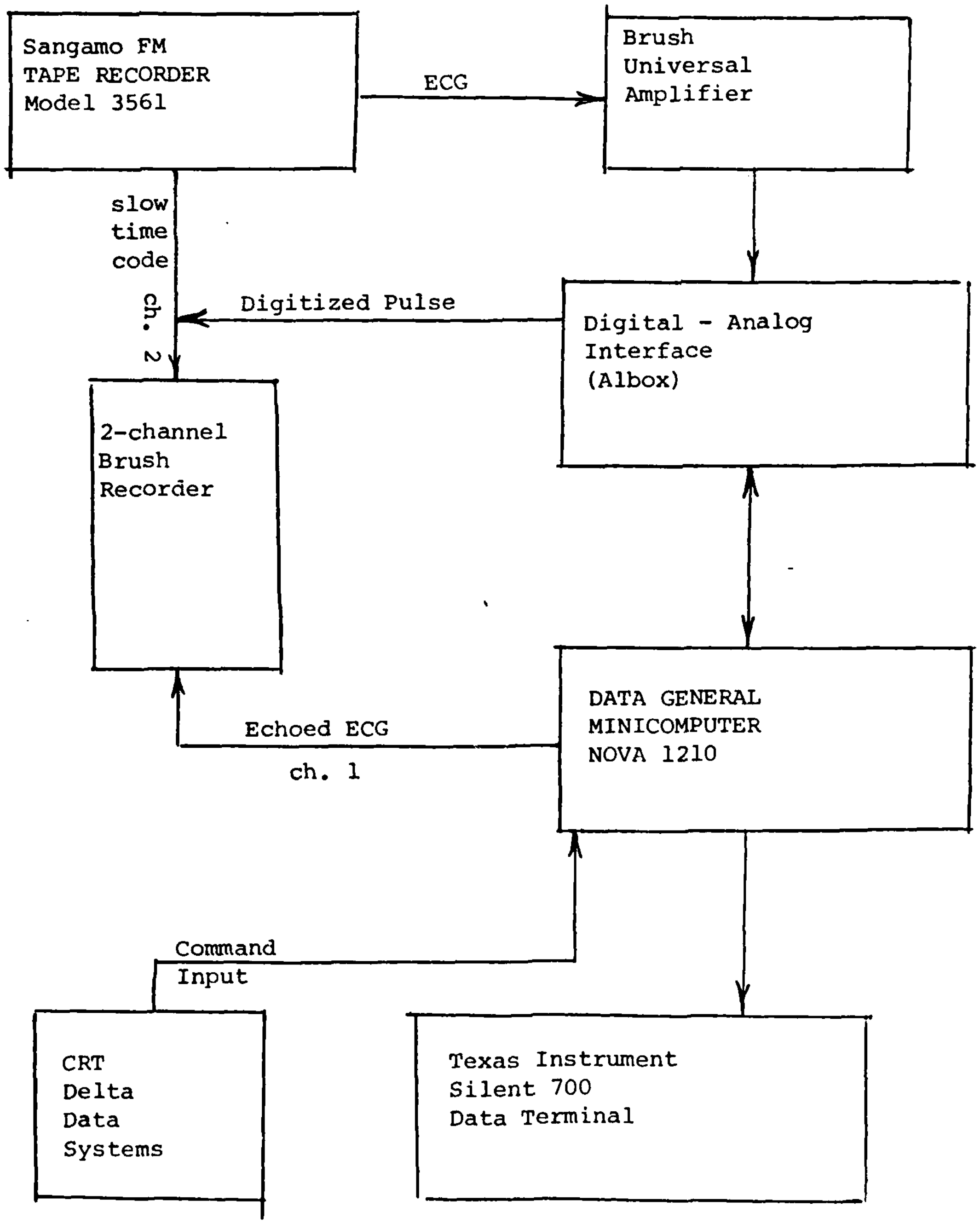

Figure 3

Block Diagram of Data Gathering System

volts and processed by a custom built interface prior to entrance into the Nova 1210 digital computer. The analog to digital conversion was performed by the Nova 1210 minicomputer. The output of the computer consisted of elapsed time, R-to-R interval, printed on a Texas Instruments Silent 700 output unit equipped with digital cassette recording format. The digital output from the Nova computer consisted of a string of R-to-R intervals and were recorded on a digital cassette. These tapes were subsequently edited by a interactive digital computer program on the Xerox data system SDS 930 computer (Appendix A). After these data were visually and manually edited to remove obvious errors, the R-to-R interval data were processed by program MINDATA into a format compatible with our classification program.

BIBLIOGRAPHY

1. Aldredge, J.L., Welch, A.J.: Variations of heart rate during sleep as a function of the sleep cycle. Electroenceph.Clin. Neurophysiol., 35: 193-198, 1973.

2. Aldredge, J.L., Welch, A.J., Richardson, P.C., Vogt, F.B.: The extraction of sleep information from heart rate data: analysis of the sleep cycle. Tech. Rep. 107, Electronics Research Center, University of Texas at Austin, Austin, Texas, 1971.

3. Aserinsky, E.: Physiological activity associated with segments of the rapid eye movement period. Chap. 16, Sleep and Altered States of Consciousness, Vol. 45, Baltimore, The Williams & Wilkins Co., 1967.

4. Aserinsky, E.: Rapid eye movement density and pattern in the sleep of normal young adults. Psychophysiol., 8: 361-375, 1971.

5. Aserinsky, E., Kleitman, N.: Two types of ocular motility occurring in sleep. J. Applied Physiol., 8:1-10, 1955.

6. Aserinsky, E.: Periodic respiratory patterns occurring in conjunction with eye movements during sleep, Science, 150:763, 1965.

7. Baust,W., Bohnert, B.: The regulation of heart rate during sleep. Exp. Brain Res., 7: 169-180, 1969.

8. Baust, W., Gagel, J.: The development of periodicity of heart rate and respiration during sleep in newborn babies. Neuropaediatrie, 8(4) 387-96, 1977.

9. Brezinova, V.: Sleep cycle content and sleep cycle duration. Electroencephal.Clin.Neurophysiol.,36:275-282, 1974.

10. Bond, W.C., Bohs, C., Ebey, J., Jr., Wolf, S.: Rhythmic heart rate variability (sinus arrhythmia) related to stages of sleep. Conditional Reflex, 8:98-107, 1973.

11. Brooks, C. McC, Hoffman, B.F., Suckling, E.E., Kleyntjens, F., Koenig, E.H., Coleman, K.S., Treumann, H.J.: Sleep and variations in certain functional activities accompanying cyclic changes in depth of sleep. J. Applied Physiol., 9:97-104, 1956.

12. Broughton, R.J., Poire, R., Tassinari, C.A.: The electrodermogram (Tarchanoff effect) during sleep. Electroencephal.Clin. Neurophysiol., 18: 691-708, 1965.

13. Carli, G.: Blood pressure and heart rate in the rabbit during animal hypnosis. Electroencephalog. Clin. Neurophysiol., 37:231-237, 1974.

14. Clausen, J., Sersen, E.A., Lidsky, A.: Variability of sleep measures in normal subjects. Psychophysiol. 11:509-516, 1974.

15. Coccagna, G., Mantovani, M., Brignani, F., Manzini, A., Lugaresi, L., Arterial pressure changes during spontaneous sleep in man. Electroencephalog. Clin. Neurophysiol., 31:277, 1971.

16. Dement, W.C., Kleitman, N.: Cyclic variations in EEG during sleep and their relation to eye movements, body motility, and dreaming. Electroencephal. Clin. Neurophysiol., 9:678-690, 1957.

17. Dement, W.C.: Some Must Watch While Some Must Sleep. San Francisco, W.H. Freeman & Co., 1974.

18. Dement, W.C.: Recent studies on the biological role of rapid eye movement sleep. Amer. J. Psychia., 122: 404-408, 1965.

19. Dement, W.C., Kleitman, N.,: The relation of eye movements during sleep to dream activity: an objective method for the study of dreaming. J. Exp. Psychol., 53:339, 1957.

20. Feinberg, I.: _Eye movement activity during sleep and intellectual function in mental retardation. Science, 159:1256, 1968.

21. Feinberg, I., Carlson, V.R.: Sleep variables as a function of age in man. Arch. Gen. Psychia., 18:239, 1968.

22. Gaillard, J.M., Tissot, R.: Principles of automatic analysis of sleep records with a hybrid system. Computers Bio-Med. Res., 6:1-13, 1973.

23. Harper, R.M., Sclabassi, R.J., Estrin, T.: Time series analysis and sleep research. IEEE Trans. Auto. Control, 19:932-942, 1974.

24. Harper, R.M., et al: Polygraphic studies of normal infants during the first six months of life, I. Heart rate and variability as a function of state, Pediatric Research, 10:945-951, 1976.

25. Hartmann, E.: The Biology of Dreaming. Springfield, Ill., Charles C. Thomas, 1967.

26. Hobson, J.A.: Sleep: physiologic aspects. New Eng. J. Med., 281:1343-1345, 1969.

27. Horne, J.A.: The effect of sleep deprivation in heart rate and respiration rate. Experientia, 33(9) 1175-77,1977.

28. Itil, T.M., Shapiro, D.M., FInk, M., Kassebaum, D.: Digital computer classifications of EEG sleep stages. Electroenceph. Clin. Neurophysiol., 27:76-83, 1969.

29. Johns, M.W.: Methods of assessing human sleep. Arch. Internal Med., 127:484, 1971.

30. Johns, T.G., et al.: Automated analysis of sleep in the rat. Electroenceph. Clin. Neurophysiol., 43(1) 103-5 1977.

31. Johnson, L.C.: Are stages of sleep related to waking behavior? Amer. Scientist, 61:326-338, 1973.

32. Jordan, J.E., Grice, T.: Bigeminy related to REM sleep. Arch. Neurol., 3(5)/319-320, 1977.

33. Khatri, I.M., Freis, E.D.: Hemodynamic changes during sleep. J. Applied Physiol., 22:867-873, 1967.

34. Kleitman, N.: Patterns of dreaming. Scientific Amer., Nov., 1960.

35. Kleitman, N.: Sleep and Wakefulness. Rev. Ed., Chicago, Univ. of Chicago Press, 1963.

36. Knott, J.R., Gibbs, F.A., Henry, C.E.: Fourier transforms of the electroencephalogram during sleep. J. Exp. Psychol., 31:465, 1942.

37. Krause, A.N., Solomon, G.E., Auld: Sleep State, apnea and bradycardia in pre-term infants. Devel. Med. Child. Neurol., 19(2)/160-168, 1970.

38. Larsen, L.E., Walter, D.O.: On automatic methods of sleep staging by EEG spectra. Electroenceph. Clin. Neurophysiol., 28:459-467, 1970.

39. Lisenby, M.J., Richardson, P.C.: The heart beat domain: a useful tool for automated classification of sleep patterns. Paper presented to the 11th Symposium on Biomathematics and Computer Science in the Life Sciences, April 3-5, 1975.

40. Lisenby, M., et al: "Sleep Wakefulness from Heart Rate Data", Tech.Rep.#173, Electronics Research Center, University of Texas at Austin, Austin, Texas, June 15, 1975

41. Lisenby, M., et al: "Sleep Wakefulness from Heart Rate Data", Tech.Rep.,Bio-Med.Electronics Research Lab.,University of Texas at Austin, Austin, Texas, May 30, 1977.

42. Lubin, A., Johnson, L.C., Austin, M.T.: Discrimination among states of consciousness using EEG spectra. Psychophysiol., 6: 122-131, 1969.

43. Miller, J.C., Horvath, S.M.: Cardiac output during sleep at altitude. Aviat. Space Environ. Med., 48(7) 621-24, 1977.

44. Orr, W.C., Hoffman, H.J.: A 90-min cardiac biorhythm: methodology and data analysis using modified periodograms and complex demodulation. IEEE Trans. Bio-Med. Eng., 21:130-143, 1974.

45. Parmelee, A.H., Akiyama, Y., Schutte, F.J.: Power spectral analysis of the EEG in newborn infants during sleep. Society Proceedings, Electroencephal.Clin.Neurophysiol., 23: 81-82, 1967.

46. Pena, A. de la., Zarcone, V., Dement, W.C.: Correlation between measures of the rapid eye movements of wakefulness and sleep. Psychophysiol., 10:488-500, 1973.

47. Radvany, M.F., Morel-Kahn, F.: Sleep and heart rate variations in premature and full-term babies. Neuropaediatrie, 7:202-312, 1976.

48. Rechtschaffen, A., Kales, A. (EDS): A manual of standardized terminology, technique, and scoring systems for sleep stages of humna subjects. Bethesda, Maryland. HEW Neurological Information Network, 1968.

49. Reite, M., et al.: Sleep in infant monkeys: Normal values and behavioral correlates. Physiology and Behavior. 16:245-51, 1976.

50. Richardson, P.C., et al: "Sleep Wakefulness from Heart Rate Data:, Tech. Rep., Bio-Medical Electronics Research Laboratory, University of Texas at Austin, Austin, Texas, May 30, 1977.

51. Smolensky, M.H., Tatar, S.E., Bergman, S.A., et al: Circadian rhythmic aspects of human cardiovascular function: a review by chronobiologic statistics. Chronobiologia, 3(4)/337-371, 1976.

52. Snyder, F., Hobson, J.A., Morrison, D.F., Goldfrank, F.: Changes in respiration, heart rate and systolic blood pressure in human sleep. J.Applied Physiol., 19: 417-422, 1964.

53. Snyder, F.: Autonomic nervous system manifestations during sleep and dreaming. Chap. 20, Sleep and Altered States of Consciousness, Vol. 45, Baltimore, The Williams & Wilkins Co., 1967.

54. Spreng, L.F., Johnson, L.C., Lubin, A.: Autonomic correlates of eye movement bursts during stage REM sleep. Psychophysiol., 4:311-323, 1968.

55. Taub, J.M., Berger, R.J.: Sleep stage patterns associated with acute shifts in the sleep-wakefulness cycle. Electroenceph.Clin.Neurophysiol., 35:613-619, 1973.

56. Watanabe, K., Iwase, K., Hara, K.: Heart rate variability during sleep and wakefulness in low-birthweight infants. Bio. of the Neonate, 22:87-98, 1973.

57. Webb, W.B., Agnew, H.W.: Sleep cycling within twenty-four hour periods. Exp. Psychol., 74:158-160, 1967.

58. Weber, F.J., Welch, A.J., Vogt, F.B., Richardson, P.C.: Detection of REM, 1 sleep stage and eye movement from beat-to-beat heart rate. Tech. Rep. 107, Electronics Research Center, University of Texas at Austin, Austin, Texas, 1973.

59. Welch, A.J., Richardson, P.C.: Computer sleep stage classification using heart rate data. Electroenceph. Clin. Neurophysiol., 34:145-152, 1973.

60. Welch, A.J.: Period analysis of space flight EEG. Aerospace Med., 42:601-606, 1971.

61. Welch, A.J., Richardson, P.C., Thomas, C.W., Aldredge, J.L.: Bandwidth reduction of sleep information. Tech. Rep. 92, Electronics Research Center, University of Texas at Austin, Austin, Texas, 1970.

62. Welch, A.J., Richardson, P.C., Thomas, C.W., Aldredge, J.L.: Final report: bandwidth reduction of sleep information. Tech. Rep. 115, Electronics Research Center, University of Texas at Austin, Austin, Texas, 1971.

63. Williams, H.L.: The problem of defining depth of sleep. Chap. 13, Sleep and Altered States of Consciousness, Vol. 45, Baltimore, The Williams & Wilkins Co., 1967.

64. Williams, R.L., Agnew, H.W., Webb, W.B.: Sleep patterns in young adults: an EEG study. Electroenceph. Clin. Neurophysiol., 17:376-381, 1964.

<u>REFEREED JOURNAL PUBLICATIONS RESULTING FROM OUR HEART RATE SLEEP STUDIES</u>

1. Welch, A.J.: Period analysis of space flight EEG. <u>Aerospace Med.</u>, 42:601-606, 1971.

2. Aldredge, J.L., Welch, A.J.: Variations of heart rate during sleep as a function of the sleep cycle. Electroenceph. Clin. Neurophysiol., 35: 193-198, 1973.

3. Welch, A.J., Richardson, P.C.: Computer sleep stage classification using heart rate data. <u>Electro-enceph. Clin. Neurophysiol.</u>, 34: 145-152, 1973.

4. Lisenby, M.J., P.C. Richardson, A.J. Welch: "Detection of Cyclic Sleep Phenomena Using Instantaneous Heart Rate", <u>Electroenceph. Clin. Neurophysiol.</u>, 40(1976) 169-177.

5. Welch, A.J.: "Computer Aided Classification of Data", <u>New Dynamics of Preventive Medicine</u>, 3:133-45,

6. Welch, A.J., P.C. Richardson, J.N. Mockford: Classification of Sleep Stage with Period Analysis Features Derived from EEG. <u>Aerospace Med.</u>, 49(2)409-14,1978.

7. Richardson, P.C., Lisenby, M.J.: "The Beatquency Domain-A Novel Application of the Fast Fourier Transform". IEEE Trans. BME 24(4): 405-8, July 1978.

ORAL PRESENTATIONS OF PROFESSIONAL MEETINGS

1. Lisenby, M.J., Richardson, P.C.: The heart beat domain: a useful tool for automated classification of sleep patterns. Paper presented to the 11th Symposium on Biomathematics and Computer Science in the Life Sciences, April 3-5, 1975.

2. Richardson, P.C., A.J. Welch, M.J. Lisenby: The heart beat domain: a useful tool for automated classification of sleep patterns. Paper presented at the 28th ACEMB, New Orleans, 1975.

3. Richardson, P.C., A.J. Welch: Detection of sleep cycle information using beat to beat heart rate. Paper presented to the Region V IEEE Conference, Austin, Texas, 1976.

4. Richardson, P.C., A.J. Welch, R.J. Montgomery: Feature extraction for microprocessor determination of sleep information from heart rate data. Paper presented at 12th Annual Meeting of AAMI, San Francisco, March 1977.

5. (Poster Session): Determination of sleep stage from heart rate data. ACEMB, Los Angeles, November 1977.

TECHNICAL REPORTS RESULTING FROM THIS STUDY

1. Aldredge, J.L., Welch, A.J., Richardson, P.C., Vogt, F.B.: The extraction of sleep information from heart rate data: analysis of the sleep cycle. Tech. Rep. 107, Electronics Research Center, University of Texas at Austin, Texas, 1971.

2. Weber, F.J., Welch, A.J., Vogt, F.B., Richardson, P.C.: Detection of REM, 1 sleep stage and eye movement from beat-to-beat heart rate. Tech. Rep. 107, Electronics Research Center, University of Texas at Austin, Austin, Texas, 1973.

3. Welch, A.J., Richardson, P.C., Thomas, C.W., Aldredge, J.L.: Bandwidth reduction of sleep information. Tech. Rep. 92, Electronics Research Center, University of Texas at Austin, Austin, Texas, 1970.

4. Welch, A.J., Richardson, P.C., Thomas, C.W., Aldredge, J.L.: Final report: bandwidth reduction of sleep information. Tech. Rep. 115, Electronics Research Center, University of Texas at Austin, Austin, Texas, 1971.

5. Lisenby, M., P.C. Richardson, A.J. Welch, Martin Nitzberg, "Sleep Wakefulness from Heart Rate Data", Tech. Rep. 173, Electronics Research Center, University of Texas at Austin, Austin, Texas, June 15, 1975.

6. Lisenby, J.J., T.P. Daubek, P.C. Richardson, A.J. Welch: "Sleep Wakefulness from Heart Rate Data", Tech. Rep. Bio-Medical Electronics Research Laboratory, University of Texas at Austin, Austin, Texas, March 30, 1976.

7. Richardson, P.C., A.J. Welch, T.P. Daubek: "Sleep Wakefulness from Heart Rate Data", Tech. Rep., Bio-Medical Electronics Research Laboratory, University of Texas at Austin, Austin, Texas, May 30, 1977.

APPENDIX A

HEART RATE DATA EDITOR

User Documentation and Code

Table of Contents

Introduction

Securing analog heart rate data free from noise is quite difficult in long term recording. In a digitizing process such as R-to-R interval recognition this noise exhibits itself as sharp spikes and is often recognized by the digitizing program as R-waves. Consequently, it is placed in the digitized data as small, but valid data values. Conversely, infeasibly large data values may also appear in the digitized data; these are due to weak ECG signals which are not detected by the digitizing program. To combat these errors a method was needed to find and correct invalid data points with a minimum amount of error added to the problem. Thus, this program, the Heart Rate Data Editing Program, with the use of the Xerox 930 graphics terminal was written to search the data for incorrect values, alert the user when one was found and allow him to make corrections.

The editor gives the user several basic functions in which to manipulate the data. Initially, the user may choose what per cent difference is acceptable between data points and how many lines of data he wishes to view on the screen. With the aid of the terminal's function keys, the user may view all the lines of data he has read in or divide the data into groups of five lines only to avoid the incessant blinking of the scope's refresh routine when too many points are exhibited on the screen at once. For correctional purposes, the user may divide up a data point which is too large or add points together which are too small using the terminal's light pen. The user may go back and look at up to ten previously processed lines of data, call for more lines of data to be read in, go to the next file or stop the program completely.

Tapes

This program calls for two, seven track magnetic tapes, the digitized data tape for input and a tape for corrected output. The input data needs to be in card image with ten, four to five digit numbers per image (10I5 format) recorded at 800 BPI. Individual files need to be separated by one end-of-file marker with the last file having two end-of-files. The output data will also be written in card image, 10I5 format. A heading, "REC i", i=1,2,3,...n, will preceed each file. Except for two end-of-files on the end of the last file, no intermediate end-of-files will written on the tape.

Keyboards

Alphanumeric Keyboard...

The scope's main keyboard and screen is basically that of any CRT device. What the user types in is displayed in the lower left hand corner as he types it in. If a mistake is made before the carriage return is given, the delete key, "DEL", on the upper left hand side will delete the entire line typed in. The carriage return key,"CR", is located on the middle right hand side of the keyboard. Once the carriage return is given, what the user types in is removed from the screen.

Function Keyboard...

To the right of the scope's alphanumeric keyboard is the function keyboard:

GO	STOP	↑	↓
NXT	BELL	←	→
A	B	C	D
ACK	L	R	ESC

For this program these keys have been assigned as follows:

GO	DISPLAY	CLEAR	THIRDS
DIVIDE	ADD	BACK	FORWARD
NEXT	STOP		

Initialization

Upon execution, the following message will be displayed on the scope.

```
HEART RATE DATA EDITING PROGRAM
     PRESS KEY "GO" TO BEGIN
```

The computer then waits for a response from the user. Upon pressing the "GO" function key, the program will display

```
MAXIMUM PER CENT DIFFERENCE ACCEPTABLE=
```

and then waits for the user to type in any free formatted number. Once it recieves it, the computer exhibits this number(N)

```
MAXIMUM PER CENT DIFFERENCE=N
     IS THIS CORRECT(Y OR N):
```

in case a wrong value was inadvertently typed in. The program will wait until it recieves a "Y" or "N". Upon a negative response, it will loop back and exhibit the editing program's title and wait for a "GO" function command. Upon a yes, it will store this per cent difference value and go on to seek the number of lines the user wishes ~~th~~ to read in. This difference value can not be changed without stopping the program.

Line Request

The program will ask

```
HOW MANY LINES OF DATA
     (10 LINES MAX) DO YOU WISH TO SEE:
```

It will wait for a number greater than zero, but less than or equal to ten. This is the actual number of lines of data read off the data tape. The data is not displayed without a function command or an error found amongst the data values.

Time

An approximation of the minute containing the R-to-R values currently being displayed is calculated from the sum of all the data values read in and is written in the upper right hand corner of the screen as "i M", i=1,2,...n.

Error Calculations

To find digitizing errors, the program will calculate the per cent difference between every two data points in the current data. If a difference is found to be greater than the maximum accepted difference value, the program will quickly try to determine which data point of the two is in error and encase it in a rectangle to alert the user. At times, it will be difficult for the machine to determine which is the incorrect point and, thus, a point which is valid when considering surrounding points may be marked as an error. At all times the user must make the decision to change a point, otherwise, marked or not, the point will remain in the data as it is.

If several errors are found, the program will display, at most, five lines of data--two lines preceeding the line containing the first incorrect data value and two lines after the line containing the error. However, if more than five lines were requested, all the lines may be viewed with errors marked by giving the DISPLAY command once the function prompt is given (see below). If no errors are found in the current data, the program will exhibit the message

```
NO ERRORS DETECTED
```

and go on to request a function command.

Maximum and Minimum

Every time all the current data is displayed, two points will be encased by diamonds. These are the minimum and maximum data values of this particular data. If only a fraction of the requested data is exhibited, minimum and maximum points may not be marked if they are not in the few lines specified.

End of Files
End of Program

When an end-of-file is encountered while reading in data, the program displays the message

```
END OF DATA
```

in the left hand corner of the screen. The last data value is found and the program resets variables accordingly for the function commands. The user is expected to correct the data, if necessary, and give either the NEXT or the STOP command before reading in more data (the FORWARD command). Failure to use one of these two commands results in the individual files read in to be tacked together. If the NEXT command is used, and no other file exists on the input tape, the second end-of-file is read in and the program promptly ends with an

```
END OF EDITING PROGRAM
```

message for the user.

Function Commands

A prompt for a function command is simply

```
FUNCTION:
```

written in the upper left hand corner of the screen.

GO...

GO is used only in the initialization routine to start the program. A message,

```
THIS FUNCTION CAUSES THE PROGRAM TO WAIT FOR
    FURTHER INSTRUCTIONS
```

is given to the user.

DISPLAY...

DISPLAY exhibits all the lines of data most recently requested. It includes the computer-found incorrect points encased by rectangles, the minimum and maximum values encased by diamonds and the lines containing the fiftieth, hundredth, and hundred and fiftieth elements of the current data marked by a ">" on the left hand side of the line.

CLEAR...

CLEAR removes everything from the screen except the function prompt, the number of the minute in which the user is currently involved, and an END OF DATA message, if necessary.

THIRDS...

THIRDS allows the user to split the data into five line groups in order to view it more closely without the continual blinking caused by the scope's refresh routine. The program will ask

```
DO YOU WISH TO SEE THE FIRST (TYPE 1),
    SECOND (TYPE 2), OR THIRD (TYPE 3) 5 LINES
    OF THE DATA:
```

If, for instance, only five lines were originally requested and the user asked to see the second or third five lines of the data, the program would continue to display the above request until it got the proper response of one. Otherwise, the data will be grouped into lines 1-5, 6-10 or lines 11-15.

DIVIDE...

DIVIDE splits a large data value into smaller values. The program will first request the user to

```
DETECT NUMBER TO BE DIVIDED
```

To do this, the user must hold the light pen on the screen, aim, and "shoot" the incorrect data value (N at location xx). A "cross" will be placed on the value "hit" and the program will do the following error checking.

```
ARRAY ELEMENT= xx
IS THIS CORRECT (Y OR N):
```

The user must check the data value's location in the current array of data points. This is facilitated by the ">" marks on the lines containing elements 41-50, 91-100, and 141-150. If the location given is not that of the desired value, the program will ask the user to

```
DETECT AGAIN
```

At times the computer will not accept values "shot" in the proximity of the "cross". In this particular case, the computer simply doesn't accept the "shot" and waits for another. Unfortunately, it doesn't give any kind of an error message to indicate it is waiting. Thus, the user needs to move the pen down and over, staying within the box, if necessary, and "shoot" again.

When the detected value is the one desired, the program will ask

```
WHAT DO YOU INTEND TO DIVIDE BY (8MAX):
```

It waits for a number greater than zero, but less than eight(M) to use as the divisor. However, if the user has inadvertantly gotten into this routine, he must nonetheless detect a number, but can type in zero at this point and be delivered back to the function prompt without disturbing the data.

The quotient and, hence, the new values (a) will be calculated and displayed along with the original value detected.

```
QUOTIENT=a
ORIGINAL VALUE SHOULD BE=N
IS THIS CORRECT(Y OR N):
```

If this value is correct, the array is expanded, these values included and control returns to the function prompt. If this is not correct, the program will prompt the user to type in the correct values with

```
WHAT M VALUES DO YOU WISH TO USE:
```

and waits for him to type M numbers in one at a time. After all the numbers are read in, the program displays them.

$$\overbrace{\text{W X Y Z}}^{\text{M numbers}}$$

```
ORIGINAL VALUE SHOULD BE= N
ARE THESE THE VALUES YOU WISH TO USE (Y OR N):
```

If these values are correct, they are included in an expanded array and control is returned to the function prompt. If these are not correct, the program loops back to the beginning of this routine without disturbing the data.

ADD...

ADD combines numbers which are too small to stand as individual data points. The program will request

```
HOW MANY NUMBERS DO YOU INTEND TO ADD
     TOGETHER (10 NUMBERS MAX):
```

and will wait for a number less than ten, but greater than zero to use as the number of addends (M). However, if the user happened into this routine unintentionally, he can leave at this point by typing in zero; this returns control to the function prompt.

Upon receiving the number of values to be added together, the program will direct the user to

```
DETECT M NUMBERS*
```

These numbers should be detected in consecutive order from left to right. To do so, the light pen should be held next to or on the screen, aimed at the desired value and "fired". The above prompt will be exhibited on the screen until the correct number of values has been detected. An error check will then be asked of the user.

NUMBERS: $\overbrace{W\ X\ Y\ Z}^{\text{M numbers}}$

```
     WILL BE ADDED TOGETHER TO GIVE A SUM=
     IS THIS CORRECT (Y OR N):
```

If these are not the correct values, the program will return to the beginning of this routine. However, if the values are correct a check is also made to determine if the numbers were detected in consecutive, ascending order by location. If they were not, the following message is displayed

```
NUMBERS NOT DETECTED IN LOCATIONALLY,
     CONSECUTIVE, ASCENDING ORDER
```

and control is again returned to the beginning of the routine.

When the desired numbers have been detected and added correctly, the data array is condensed to hold their sum and control is returned to the function prompt.

---

*More information on point detection can be found in the divide command information.

BACK...

BACK allows the user to edit up to ten lines of data which have already been processed for output. On the first read of a file, there is no previous data to concatenate with the present. Thus, the program will respond with

```
     NO NUMBERS HAVE BEEN PROCESSED TO LOOK AT
```

and return control to the function prompt. Otherwise, the program will go on to ask the user

```
     HOW MANY LINES BACK
          (10 LINES MAX) DO YOU WISH TO SEE:
```

and waits for a number greater than zero, but less than or equal to ten. However, only fifteen lines of data can be exhibited on the screen at any one time. The maximum number of data points secured from the preceding data will be diminished to allow the user to view all the current values in context of fewer previous points; five lines back is the maximum number one can view with a full screen of ten lines. Also, the program will compensate for the user asking for more lines than are stored. In this case, it will display all the lines that have been buffered for output with the current data.

When lines are secured from the previous read, they are placed in continuity with the current lines of data and the entire array is scanned for errors as if all the data had just been read in. All functions can be performed on this large array except another BACK function. When a second BACK command is given on an already expanded array, all the "old" data points are replaced in the memory buffer and the original data array is restored before the BACK routine is initiated. Therefore, the same data will be displayed on the screen as before.

FORWARD...

FORWARD stores the current, corrected data to allow the user to obtain more data to edit. It clears the screen, prepares the old buffered data for output, writes this prepared data on tape, and buffers the current data. If the BACK command was executed on the current data, FORWARD also restores the original array before buffering it.

NEXT...

NEXT obtains a new file for the user. It first error checks to make sure the user intended to execute this command by asking

ARE YOU SURE YOU WANT THE NEXT FILE (Y OR N):

If this routine was initiated unintentionally, control returns to the function prompt. Otherwise, all remaining data in the input tape's current file is skipped, data in all memory buffers as well as the current data ~~is~~ are written on the output tape, all buffers, variables, and flags are reinitialized for a new file of data and control is returned to the function prompt. If this command was given in compliance with the

END OF DATA

message, the immediately succeeding file is the next one read and this error message is removed from the screen.

STOP...

STOP allows the user to halt the program. To protect the user from stopping unintentionally, it first asks

ARE YOU SURE YOU WANT TO STOP (Y OR N):

and control is returned to the function prompt upon a negative response. Otherwise, all previous and current data are written on the output tape and the screen is cleared except for the message

END OF EDITING PROGRAM

Conclusion

The Heart Rate Data Editing Program was written with the intention of correcting errors in digitized R-to-R interval heart rate data and was used to successfully edit several files of data. Programming a computer to find almost all possible incorrect points and correcting these points interactively, increases the amount of data wich can be edited at any one time, decreases the number of persons needed to do this job, and, therefore, decreases the amount of human error.

Appendix A

```
C
C HEART RATE EDITING PROGRAM
C      ENABLES USER TO INTERACTIVELY EDIT DIGITIZED DATA
C
C INPUT...
C      DIGITIZED DATA IN 10I5 FORMAT WITH EOF MARKINGS BETWEEN FILES
C OUTPUT...
C      CORRECTED DATA IN 10I5 FORMAT, EACH FILE PRCEEDED BY ≡REC XX≡
C
       COMMON NDATA(200),NPLOT(50),NBUF(100),KBUF(100),NTEMP(10),MBUF(100
      1)
C
C    NDATA... INTEGER ARRAY INTO WHICH DATA IS READ, EACH DATA POINT OCCUPYING
C             ONE WORD IN THE ARRAY.  OLD DATA IS WRITTEN OVER UPON EACH READ.
C
C    NPLOT ...INTEGER ARRAY HOLDING THE LOCATIONS OF POINTS  THE PROGRAM FINDS IN
C             ERROR, EACH LOCATION OCCUPYING ONE WORD IN THE ARRAY.  OLD DATA IS
C             WRITTEN OVER UPON NEW DATA READ IN
C
       COMMON/KMODAL/IOV,MODE
       COMMON/LIM/LIM1,LIM2
       COMMON/FOREGND/JCB(125),JSB(1500),LITE(20)
C INITIALIZE VARIABLES AND DATA ARRAY
       DATA NUM,LAST,NO/0,0,0/NFROM,IPT1,IPT2/1HM,0,0/NEOF/1HN/NREAD/1HN/
       DATA LSUM,NBK,NSTOP,NTITLE,KSTOP/0,0,1HN,1,1HN/NOMORE/1HN/
       IOV=1  $  MODE=1
       NBUF(1)=0  $  KBUF(1)=0  $  MBUF(1)=0
       DO 111 J=1,200
111    NDATA(J)=0
C INITIALIZE SCOPE,FORMATTED READ AND WRITE PACKAGES
       CALL SCOPE(JSB,1500)
       CALL SWRITE(IOV,0,MODE,1)
       CALL SREAD(2)
C ASK USER FOR PERCENT DIFFERENCE ACCEPTABLE
       CALL INITIAL(NPD)
C WRITE REC NO. ON OUTPUT TAPE TO DIFERENTIATE BETWEEN FILES
1      WRITE (4,2) NTITLE
2      FORMAT(*REC*,1X,I2)
C ASK USER FOR NUMBER OF DATA LINES HE WISHES TO SEE
4      CALL LINE(NFROM,NLINES,NUM)
C EOFSET SEARCHES FOR EOF AND JUMPS TO EOF ROUTINE WHEN IT FINDS EOF
       CALL EOFSET(6S,3)
C READ IN R-RDATA POINTS
3      READ(3,7)(NDATA(K),K=1,NUM)
7      FORMAT(10I5)
       NOMORE=1HN  $  NREAD=1HY
       GOTO 5
6      CALL EOF(NUM,NDATA)
       NEOF=1HY
       IF(NOMORE.EQ.1HY) GOTO 35
C TIME DISPAYS THE APPROXIMATE MINUTE BEING DISPAYED IN THE UPPER RIGHT CORNER
5      CALL TIME(NDATA,LSUM,NUM)
C FIGURE FINDS DATA POINTS ABOVE ACCEPTED PER CENT DIFFERENCE LIMIT AND
C  STORES THEIR LOCATION IN NPLOT
19     CALL FIGURE (NDATA,NUM,NPLOT,NO,NPD,LAST,KMN,KMX,NREAD)
C PROMPT USER TO CHOSE A FUNCTION KEY
20     WRITE(1,31)
31     FORMAT(1/,2X,*FUNCTION≤*)
C
C USER DEFINES FUNCTION
       GOTO (21,22,23,24,25,26,27,28,29,30) KEY(10)
```

```
C
C GO IS USED ONLY IN THE INITIALIZATION ROUTINE
21      WRITE(1,32)
32      FORMAT(4/,*THIS FUNCTION CAUSES THE PROGRAM TO WAIT FOR*,/,5X,*FUR
       1THER INSTRUCTIONS*)
        CALL DELAY(3)
        GOTO 20
C
C DISPLAY CURRENT DATA ON SCREEN
22      LIM1=1  $  LIM2=NUM
        CALL REMOVE(2)  $  CALL REMOVE(3)
        CALL WRITLIM(NDATA)
        CALL PLOTTER(KMN,KMN)  $  CALL PLOTTER(KMX,KMX)
        IF(NO.EQ.0) GOTO 39
        DO 33 J=1,NO
33      CALL PLOTTER (0,NPLOT(J))
39      GOTO 20
C
C CLEAR SCREEN
23      CALL REMOVE(2)  $  CALL REMOVE(3)
        GOTO 20
C
C THIRDS DIVIDES DATA INTO THIRDS FOR EASIER VIEWING
24      CALL REMOVE(2)  $  CALL REMOVE(3)
        CALL THIRDS (NUM,NDATA,NO,NPLOT,KMN,KMX)
        GOTO 20
C
C DIVIDE SPLITS A NUMBER WHICH IS TOO BIG
25      CALL DIVIDE(NDATA,NUM)
        NO=0  $  GOTO 19
C
C ADD COMBINES NUMBERS WHICH ARE TOO SMALL
26      CALL ADD (NDATA,NUM)
        NO=0  $  GOTO 19
C
C BACK UP PROCESSED DATA
27      IF (NBK.EQ.0) GOTO 44
        IF (NFROM.NE.1HB) GOTO 43
        CALL FORBACK(KNUM,NLOW,IPT1,IPT2,KVAL,NUM,NDATA,KBUF,NBUF,NTEMP,NV
       1AL)
43      CALL REMOVE(2)  $  CALL REMOVE (3)
C BACK RECALLS PREVIOUS LINES EXHIBITED
        CALL BACK(NDATA,NUM,IPT1,IPT2,NBUF,KBUF,NTEMP,KNUM,NLOW,NVAL,KVAL,
       1NFROM,NBK)
        NO=0  $  GOTO 19
C NO DATA PROCESSED TO LOOK AT
44      WRITE(1,45)
45      FORMAT(/,2X,*NO NUMBERS HAVE BEEN PROCESSED TO LOOK AT*)
        CALL DELAY(3)
        GOTO 20
C
C FORWARD--GET MORE DATA
28      IF (NFROM.NE.1HB) GOTO 40
        NFROM=1HM
        CALL FORBACK(KNUM,NLOW,IPT1,IPT2,KVAL,NUM,NDATA,KBUF,NBUF,NTEMP,NV
       1AL)
C FORWARD STORES AND OUTPUTS CORRECTED DATA
40      CALL FORWARD(NDATA,NUM,IPT1,IPT2,NBUF,KBUF,NTEMP,NSTOP,KSTOP,MBUF)
        CALL REMOVE(2)  $  CALL REMOVE(3)
        IF (NSTOP.EQ.1HY) GOTO 34
        IF (KSTOP.EQ.1HY) GOTO 35
        NO=0
```

```
      NBK=(NUM/10)+NBK
      GOTO 4
C
C GET NEXT FILE
29    CALL QUIT(NSTOP,1HF)
      IF(NSTOP.EQ.1HN) GOTO 20
      GOTO 28
34    CALL REMOVE(5)
      IF(NEOF.EQ.1HN) CALL SKIPF(3,1)
      NTITLE=NTITLE+1
C REINITIALIZE VARIABLES
      NOMORE=1HY  $  NEOF=1HN
      NFROM=1HM  $  IPT1=0  $  IPT2=0  $  NBK=0  $  NUM=0  $  LAST=0
      LSUM=0 $ NBUF(1)=0 $ KBUF(1)=0 $ MBUF(1)=0 $ NO=0 $ NSTOP=1HN
      GOTO 1
C
C STOP EDIT PROGRAM
30    CALL QUIT(KSTOP,1HS)
      IF (KSTOP.EQ.1HN) GOTO 20
      GOTO 28
35    WRITE(1,9)
9     FORMAT(/,*END OF EDITING PROGRAM*)
      CALL DELAY (5)
      STOP
      END
C
C
C
      SUBROUTINE INITIAL (NPD)
C
C INITIAL ASKS USER FOR ACCEPTED PER CENT DIFFERENCE LIMIT
C
C INPUT...
C   (IN COMMON)
C     IOV=SCREEN OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C     MODE=FLAG TO INDICATE WHICH FORMATED WRITE MODE TO USE
C OUTPUT...
C     NPD=MAXIMUM PER CENT DIFFERENCE BETWEEN DATA POINTS ACCEPTABLE
C
      COMMON/KMODAL/IOV,MODE
C DISPLAY TITLE ON SCREEN AND WAIT FOR USER TO START PROGRAM
1     WRITE(1,20)
20    FORMAT(1X,*HEART RATE DATA EDITING PROGRAM*,2/,5X,*8 AUGUST 1977*,
     14/,*PRESS KEY ≠GO≠ TO BEGIN*)
C WAIT FOR USER TO START PROGRAM
      N=KEY(1)
      WRITE(1,2)
2     FORMAT(5X,*MAXIMUM PER CENT DIFFERENCE ACCEPTABLE=*)
      READ (2,3) NPD
3     FORMAT(I)
C ERROR CHECK
      WRITE (1,4) NPD
4     FORMAT(5X,*MAXIMUM PER CENT DIFFERENCE=*,I5,/,5X,*IS THIS CORRECT(
     1Y OR N)≤*)
5     READ (2,6) IANS
6     FORMAT(A1)
      IF (IANS.NE.1HN.AND.IANS.NE.1HY) GOTO 5
      IF (IANS.EQ.1HN) GOTO 1
      RETURN
      END
C
C
```

```
C  . .   .
      SUBROUTINE LINE(NFROM,NLINES,NUM)
C
C LINE FINDS NUMBER OF LINES USER WISHES TO SEE
C
C INPUT...
C     NFROM=FLAG WHETHER BACK HAS BEEN CALLED OR NOT(B=YES,M=NO)
C     NLINES=NUMBER OF LINES( OR NUMBER OF LINES BACK) USER WISHES TO SEE
C     NUM=NUMBER OF ELEMENTS TO BE READ INTO ARRAY NDATA
C   (IN COMMON)
C     IOV=SCREEN OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C     MODE=FLAG TO INDICATE WHICH FORMATED WRITE MODE TO USE
C
      COMMON/KMODAL/IOV,MODE
      IF(NFROM.EQ.1HB) GOTO 3
1     WRITE(1,2)
2     FORMAT(/,*HOW MANY LINES OF DATA*,/,5X,*(10 LINES MAX) DO YOU WISH
     1 TO SEE≤*)
      GOTO 5
C OUTPUT LINE FOR BACK ROUTINE
3     WRITE(1,4)
4     FORMAT(/,*HOW MANY LINES BACK*,/,5X,*(10 LINES MAX) DO YOU WISH TO
     1 SEE≤*)
5     READ(2,7) NLINES
7     FORMAT(I)
C USER INPUT ERROR CHECK
      IF((NLINES.LE.0.OR.NLINES.GT.10).AND.NFROM.EQ.1HB) GOTO 3
      IF(NLINES.LE.0.OR.NLINES.GT.10) GOTO 1
C NUM IS NUMBER OF ELEMENST IN DATA ARRAY
      NUM=NLINES*10
      RETURN
      END
C
C
C
      SUBROUTINE EOF(NUM,NDATA)
C
C EOF TELLS USER EOF HAS BEEN FOUND AND SEARCHES FOR FIRST ZERO IN ARRAY
C
C INPUT...
C     NUM=NUMBER OF ELEMENTS USER REQUESTED TO BE IN NDATA ARRAY
C     NDATA=DATA CURRENTLY BEING EDITED
C   (IN COMMON)
C     IOV=SCREEN OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C     MODE=FLAG TO INDICATE WHICH FORMATED WRITE MODE TO USE
C OUTPUT...
C     NUM=NUMBER OF NON-ZERO VALID POINTS IN NDATA
C
      COMMON/KMODAL/IOV,MODE
      DIMENSION NDATA(200)
      IOV=5
C TELL USER THERE IS NO MORE DATA
      WRITE (1,1)
1     FORMAT(*END OF DATA*)
C FIND LAST VALID DATA VALUE
      DO 2 K=1,NUM
      IF (NDATA(K).EQ.0)  NUM=K-1  $  GOTO 3
2     CONTINUE
3     IOV=1
      RETURN
      END
      SUBROUTINE TIME(NDATA,LSUM,NUM)
C
C TIME DETERMINES WHICH MINUTE IS BEING PROCESSED
C
C INPUT...
C     NDATA=ARRAY OF CURRENT DATA POINTS BEING PROCESSED
C     LSUM=SUM OF ALL DATA POINTS PROCESSED SO FAR (NOT CURRENT POINTS)
C     NUM=NUMBER OF ELEMENTS IN NDATA
C OUTPUT...
C     LSUM=SUM OF ALL DATA POINTS PROCESSED SO FAR INCLUDING CURRENT DATA
C
      COMMON/KMODAL/IOV,MODE
      DIMENSION NDATA(200)
C ADD TOTAL NO. OF SECONDS
      DO 1 J=1,NUM
1     LSUM=NDATA(J)+LSUM
C DETERMINE PROPER MINUTE
      MIN=(LSUM/60000)+1
      CALL REMOVE(4)
      IOV=4
C DISPLAY  MINUTE
      WRITE(1,2)MIN
2     FORMAT(35X,I3,1X,*M*)
      IOV=1
      RETURN
      END
C
C
C
      SUBROUTINE PLOTTER(IVAR,I)
C
C PLOTTER PUTS SQUARES AROUND BAD DATA POINTS AND DIAMONDS AROUND MIN AND MAX
C
C INPUT...
C     IVAR=FLAG INDICATING PLOTTING DIAMONDS(.GT.0) OR SQUARES(.LE.0)
C     I=LOCATION OF DATA ELEMENT TO BE MARKED
C
      DIMENSION IXAR(5),IYAR(5)
      CALL GETXY(I,IX,IY)
      IF (IVAR.GT.0) GOTO 2
C PLOT VECTORS INTO SQUARES AROUND DATA POINTS
      IXAR(1)=IX-5   $  IYAR(1)=IY-12
      IXAR(2)=IX-5   $  IYAR(2)=IY+36
      IXAR(3)=IX+98   $  IYAR(3)=IY+36
      IXAR(4)=IX+98   $  IYAR(4)=IY-12
      IXAR(5)=IX-5   $  IYAR(5)=IY-12
3     CALL SIPLOT2(3,IXAR,IYAR,5,-6)
      RETURN
C PLOT VECTORS INTO DIAMONDS AROUND DATA POINTS
2     IXAR(5)=IX+49   $  IYAR(5)=IY-12
      IXAR(1)=IX+49   $  IYAR(1)=IY-12
      IXAR(2)=IX-8   $  IYAR(2)=IY+12
      IXAR(3)=IX+49   $  IYAR(3)=IY+36
      IXAR(4)=IX+98   $  IYAR(4)=IY+12
      GOTO 3
      END
C
C
C
```

```
      SUBROUTINE FIGURE(NDATA,NUM,NPLOT,NO,NPD,LAST,KMN,KMX,NREAD)
C
C FIGURE FINDS POINTS ABOVE ACCEPTED PER CENT DIFFERENCE LIMIT BY COMPARING
C EVERY TWO DATA VALUES
C
C INPUT...
C      NDATA=DATA WHICH NEEDS TO BE EDITED
C      NUM=NUMBER OF ELEMENTS IN ARRAY NDATA
C      NPD=ACCEPTABLE PER CENT DIFFERENCE LEVEL
C      LAST=LAST DATA VALUE OF PREVIOUS DATA ARRAY
C   (IN COMMON)
C      IOV=SCOPE OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C      MODE=INDICATES WHICH FORMATTED WRITE MODE TO USE
C OUTPUT...
C      NPLOT=LOCATAIONS OF BAD DATA POINTS IN NDATA
C      NO=NUMBER OF ELEMENTS IN ARRAY NPLOT
C      KMN=LOCATION OF MINIMUM VALUED DATA POINT IN NDATA
C      KMX=LOCATAION OF MAXIMUM VALUED DATA POINT IN NDATA
C   (IN COMMON)
C      LIM1=LOWER LIMIT↓ LOCATION IN NDATA TO BEGIN WRITING DATA VALUES FROM
C      LIM2=UPPER LIMIT↓ LOCATION IN NDATA TO TERMINATE WRITING DATA VALUES
C
      COMMON/KMODAL/IOV,MODE
      COMMON/LIM/LIM1,LIM2
      DIMENSION NDATA(200),NPLOT(50)
      MIN=90000  $  MAX=0
C CLEAR SCREEN
      CALL REMOVE(2)  $  CALL REMOVE(3)
C SET LAST TO FIRST DATA ELEMENT FOR VERY FIRST ELEMENT
      IF (LAST.EQ.0) LAST=NDATA(1)
C SERACH FOR BAD DATA POINTS
      DO 11 K=1,NUM
      TEMP1=NDATA(K)
C FIRST ELEMENT OF ARRAY IS A SPECIAL CASE
      IF (K.EQ.1) TEMP2=LAST  $  GOTO 10
C COMPARE FIRST DATA POINT IN CURRENT ARRAY TO LAST DATA POINT IN THE
C PREVIOUS ARRAY (VERY FIRST ELEMENT IS COMPARED WITH ITSELF
      TEMP2 = NDATA(K-1)
C TAKE PER CENT DIFFERENCE BETWEEN 2 POINTS
10    PDIF=ABS(((TEMP1-TEMP2)/TEMP2)*100.0)
C IF PERCENT DIFFERENCE IS GREATER THAN WHAT USER SET, SAVE LOCATION OF BAD
C POINT IN NPLOT
      IF(PDIF.GE.NPD) CALL SAVEBAD(K,NO,NPLOT)
C FIND MINIMUM AND MAXIMUM DATA VALUES
11    CALL MINMAX(K,MIN,MAX,NDATA,KMN,KMX)
C SAVE LAST DATA POINT IN NDATA ARRAY FOR COMPARISON
      IF (NREAD.EQ.1HY) LAST=NDATA(NUM)
      NREAD=1HN
C TELL USER IF NO ERRORS ARE FOUND
      IF (NO.EQ.0) IOV=3  $  WRITE(1,5)  $  GOTO 6
5     FORMAT(2/,*NO ERRORS DETECTED*)
C DISPLAY DATA WITH BAD POINTS MARKED
      CALL WRITERR (NUM,NDATA,NPLOT,NO,KMN,KMX)
6     IOV=1
      RETURN
      END
      SUBROUTINE SAVEBAD(K,NO,NPLOT)
C
C SAVEBAD SAVES LOCATIONS OF BAD DATA POINTS IN ARRAY NPLOT
C
C INPUT...
C     K=LOCATION OF POINT IN NDATA FOUND TO BE IN ERROR
C     NO=NUMBER OF BAD DATA POINTS IN NPLOT
C     NPLOT=LOCATIONS OF BAD DATA POINTS
C OUTPUT...
C     NO=NUMBER OF BAD DATA POINTS IN NPLOT
C     NPLOT=LOCATIONS OF BAD DATA POINTS
C
      DIMENSION NPLOT(50)
      J=K
C IF LOCATION OF BAD POINT IS THE FIRST OR SECOND ELEMENT IN THE ARRAY SKIP
C FURTHER TESTS AND STORE THE VALUES LOCATION
      IF (K.LE.2) GOTO 12
      IF (NO.LE.0) GOTO 12
      IF (NPLOT(NO).EQ.(K-1)) GOTO 13
      IF (NPLOT(NO).EQ.(K-2)) J=K-1
12    NO=NO+1
      NPLOT(NO)=J
13    RETURN
      END
C
C
      SUBROUTINE WRITERR (NUM,NDATA,NPLOT,NO,KMN,KMX)
C
C FIND AND SHOW 2 LINES BEFORE AND 2 LINES AFTER BAD DATA POINTS
C
C INPUT...
C     NDATA=ARRAY OF DATA POINTS CURRENTLY BEING EDITED
C     NUM=NUMBER OF ELEMENTS CURRENTLY IN DATA ARRAY TO BE EDITED
C     NPLOT=LOCATIONS OF BAD DATA POINTS THAT THE PROGRAM FOUND IN CURRENT DATA
C     NO=NUMBER OF BAD DATA POINTS IN NPLOT
C     KMN=LOCATION OF MINIMUM DATA POINT VALUE IN NDATA
C     KMX=LOCATION OF MAXIMUM DATA POINT VALUE IN NDATA
C   (IN COMMON)≤
C     IOV=SCOPE OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C     MODE=INDICATES WHICH FORMATTED WRITE MODE TO USE
C     LIM1=LOWER LIMIT↓ LOCATION IN NDATA TO BEGIN WRITING DATA VALUES FROM
C     LIM2=UPPER LIMIT↓ LOCATION IN NDATA TO TERMINATE WRITING DATA VALUES
C
      COMMON/KMODAL/IOV,MODE
      COMMON/LIM/LIM1,LIM2
      DIMENSION NDATA(200),NPLOT(50)
C CLEAR SCREEN
      CALL REMOVE(3)  $  CALL REMOVE(2)
C FIND 2 LINES BEFORE AND 2 LINES AFTER BAD DATA POINTS
      CALL LIMITS(NPLOT,NUM,1)
C SHOW 2 LINES BEFORE AND 2 LINES AFTER BAD DATA POINT
      CALL WRITLIM(NDATA)
C PLOT MIN, MAX POINTS IF THEY ARE WITHIN LIMITS
      IF (KMN.GE.LIM1.AND.KMN.LE.LIM2) CALL PLOTTER (KMN,KMN)
      IF (KMX.GE.LIM1.AND.KMX.LE.LIM2) CALL PLOTTER (KMX,KMX)
C PLOT BAD POINTS WITHIN LIMIT
      DO 3 J=1,NO
      IF(NPLOT(J).GE.LIM1.AND.NPLOT(J).LE.LIM2)CALL PLOTTER(0,NPLOT(J))
3     CONTINUE
      RETURN
      END
      SUBROUTINE MINMAX(K,MIN,MAX,NDATA,KMN,KMX)
C
C MINMAX DETERMINES THE MINIMUM AND MAXIMUM DATA POINTS
C
C INPUT...
C     K=ELEMENT POINTER IN NDATA
C     MIN=RELATIVE MINIMUM DATA VALUE FROM POINT TO POINT
C     MAX=RELATIVE MAXIMUM DATA VALUE FROM POINT TO POINT
C     NDATA=ARRAY OF DATA POINTS TO BE EDITED
C OUTPUT...
C     KMN=MINIMUM NUMBER AMONGST ALL DATA VALUES ENCOUNTERED IN ARRAY
C     KMX=MAXIMUM NUMBER AMONGST ALL DATA VALUES ENCOUNTERED IN ARRAY
C     MIN=RELATIVE MINIMUM DATA VALUE IN NDATA FROM POITN TO POINT
C     MAX=RELATIVE MAXIMUM DATA VALUE FROM POINT TO POINT
C
      DIMENSION NDATA(200)
      MIN=MIN0(MIN,NDATA(K))
      IF(MIN.EQ.NDATA(K))KMN=K
      MAX=MAX0(MAX,NDATA(K))
      IF (MAX.EQ.NDATA(K)) KMX=K
      RETURN
      END
C
C
C
      SUBROUTINE LIMITS(NPLOT,NUM,K)
C
C LIMITS FINDS 2 LINES BEFORE AND 2 LINES AFTER BAD DATA POINT
C
C INPUT...
C     NPLOT=LOCATIONS OF BAD DATA POINTS FOUND IN CURRENT DATA
C     NUM=NUMBER OF ELEMENTS IN NDATA
C     K=LOCATION TO FIND LIMITS BEFORE AND AFTER
C OUTPUT...
C    (IN COMMON)
C     LIM1=LOWER LIMIT↓ LOCATION IN NDATA TO BEGIN WRITING DATA VALUES FROM
C     LIM2=UPPER LIMIT↓ LOCATION IN NDATA TO TERMINATE WRITING DATA VALUES
C
      COMMON/LIM/LIM1,LIM2
      DIMENSION NPLOT(50)
      ILINES=NPLOT(K)/10
      LT=MOD(NPLOT(K),10)
      IF(LT.EQ.0)ILINES=ILINES-1
C LIM1 IS LOWER LIMIT
      LIM1=((ILINES-1)*10)-9
C LIM2 IS UPPER LIMIT
      LIM2=(ILINES+3)*10
      IF(LIM1.LE.0)LIM1=1
      IF(LIM2.GT.NUM)LIM2=NUM
      RETURN
      END
      SUBROUTINE GETXY(I,IX,IY)
C
C GET XY DETERMINES X,Y COORDINATES FROM ARRAY LOCATION
C
C INPUT...
C     I=LOCATION OF DATA POINT IN DATA ARRAY
C OUTPUT...
C     IX=X-COORDINATE OF DATA POINT ON SCREEN
C     IY=Y-COORDINATE OF DATA POINT ON SCREEN
C
      ILINES=I/10
      MUL=MOD(I,10)
      IF(MUL.EQ.0) MUL=10  $  ILINES=ILINES-1
      IF(I.EQ.1) IX=36  $  IY=753  $  GOTO 35
      IX=(MUL*90)-54
      IY=753-(ILINES*48)
35    RETURN
      END
C
C
C
      SUBROUTINE WRITLIM(NDATA)
C
C SHOW 2 LINES BEFORE AND 2 LINES AFTER BAD DATA POINT
C
C INPUT...
C     NDATA=ARRAY OF CURRENT DATA VALUES
C   (IN COMMON)
C     IOV=SCOPE OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C     MODE=INDICATES WHICH FORMATTED WRITE MODE TO USE
C     LIM1=LOWER LIMIT↓ LOCATION IN NDATA TO BEGIN WRITING DATA VALUES FROM
C     LIM2=UPPER LIMIT↓ LOCATION IN NDATA TO TERMINATE WRITING DATA VALUES
C
      COMMON/LIM/LIM1,LIM2
      COMMON/KMODAL/IOV,MODE
      DIMENSION NDATA(200),KHAR(2),KARRAY(13)
      KHAR(1)=1H=  $  NCHAR=52
      CALL REMOVE(2)   $  CALL REMOVE(3)
C GET COORDINATES OF LOWER LIMIT
      CALL GETXY(LIM1,IX,IY)
      JVAL=LIM1-1
      JTEMP=(LIM2-LIM1)+1
      NTIMES=JTEMP/10
      IF (MOD(JTEMP,10).NE.0) NTIMES=NTIMES+1 $ NSUB=MOD(JTEMP,10)*5
      IF (MOD(NSUB,4).NE.0) NSUB=NSUB+(4-MOD(NSUB,4))
C GRAPH NUMBERS ON SCOPE
      DO 4 K=1,NTIMES
      ENCODE (52,3,KARRAY)(NDATA(J+JVAL),J=1,10)
3     FORMAT(10I5,2X)
      IF (K.EQ.NTIMES) NCHAR=NSUB
      CALL STITLE2 (2,IX,IY,18,0,KARRAY,NCHAR,0)
      IY=IY-48  $  IX=36  $  JVAL=JVAL+10
4     CONTINUE
C DISPLAY CHARACTER ,=, MARKING LINE NO. 5
      CALL STITLE(2,18,562,KHAR,1,0)
      CALL STITLE(2,18,322,KHAR,1,0)
      IOV=1
      RETURN
      END
C
C
```

```
      SUBROUTINE THIRDS(NUM,NDATA,NO,NPLOT,KMN,KMX)
C
C THIRDS EXHIBITS ONLY THE THIRD OF THE DATA THE USER WISHES TO SEE
C
C INPUT...
C     NUM=NUMBER OF ELEMENTS IN NDATA
C     NDATA=ARRAY OF CURRENT DATA POINTS BEING EDITED
C     NO=NUMBER OF BAD POINTS IN NPLOT
C     NPLOT=LOCATIONS OF BAD DATA VALUES
C     KMN=LOCATION OF DATA POINT WITH THE MINIMAL VALUE OF THE CURRENT DATA
C     KMX=LOCATION OF DATA POINT WITH THE MAXIMAL VALUE OF THE CURRENT DATA
C   (IN COMMON)
C     IOV=SCOPE OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C     MODE=INDICATES WHICH FORMATTED WRITE MODE TO USE
C     LIM1=LOWER LIMIT↓ LOCATION IN NDATA TO BEGIN WRITING DATA VALUES FROM
C     LIM2=UPPER LIMIT↓ LOCATION IN NDATA TO TERMINATE WRITING DATA VALUES
C
      COMMON/KMODAL/IOV,MODE
      COMMON/LIM/LIM1,LIM2
      DIMENSION NDATA(200),NPLOT(50)
18    WRITE(1,12)
12    FORMAT(/,2X,*DO YOU WISH TO SEE THE FIRST (TYPE 1), *,/,5X,*SECOND
     1 (TYPE 2), OR THIRD (TYPE 3) 5 LINES*,/,5X,* OF THE DATA≤*)
19    READ(2,17)NTHIRD
17    FORMAT(I)
      IF (NTHIRD.NE.1.AND.NTHIRD.NE.2.AND.NTHIRD.NE.3)GOTO 18
C FIND LIMITS FOR WHICH THIRD OF DATA BUFFER USER WISHES TO SEE
      GOTO (13,14,15)NTHIRD
13    NFAR=MIN0(NUM,50)
      LIM1=1  $  LIM2=NFAR  $  GOTO 16
14    IF (NUM.LE.51) GOTO 18
      NFAR=MIN0(NUM,100)
      LIM1=51  $  LIM2=NFAR  $  GOTO 16
15    IF (NUM.LE.101) GOTO 18
      NFAR=MIN0(NUM,150)
      LIM1=101  $  LIM2=NFAR
C DISPLAY LIMITED DATA ON SCREEN
16    CALL WRITLIM(NDATA)
      IF (NO.EQ.0) GOTO 21
C DISPLAY BAD DATA POINTS WITHIN LIMITS
      DO 20 I=1,NO
      IF(NPLOT(I).GE.LIM1.AND.NPLOT(I).LE.LIM2) CALL PLOTTER(0,NPLOT(I))
20    CONTINUE
21    IF (KMN.GE.LIM1.AND.KMN.LE.LIM2) CALL PLOTTER(KMN,KMN)
      IF (KMX.GE.LIM1.AND.KMX.LE.LIM2) CALL PLOTTER(KMX,KMX)
      RETURN
      END
      SUBROUTINE DIVIDE (NDATA,NUM)
C
C DIVIDE SPLITS DATA POINTS WHICH ARE TOO BIG
C
C INPUT...
C     NDATA=ARRAY OF DATA POINTS CURRENTLY BEING EDITED CONTAINING POINT USER
C                WISHES TO DIVIDE INTO SEVERAL NUMBERS
C     NUM=NUMBER OF ELEMENTS IN NDATA
C OUTPUT...
C     NDATA=ARRAY OF CURRENT DATA POINTS WITH BAD POINT DIVIDED INTO SEVERAL
C           NUMBERS
C     NUM=NO. OF ELEMENTS IN NDATA WITH POINT HAVING BEEN DIVIDED OUT
C
      COMMON/FOREGND/JCB(125),JSB(1500),LITE(20)
      COMMON/LIM/LIM1,LIM2
      DIMENSION NDATA(200),LTEMP(10)
      WRITE(1,10)
10    FORMAT (/,*DETECT NUMBER TO BE DIVIDED*)
C DETECT ARRAY LOCATION OF BAD POINT FRM LIGHT PEN
      CALL DETECT(KEL)
C ASK USER WHAT HE WISHES TO DIVIDE BY
      CALL DIVNUM(NDATA,KEL,MDIV,LTEMP)
C DIVIDING BY  NR6 INDICATES NEED YO LEAVE ROUTINE
      IF (MDIV.EQ.0) GOTO 18
C EXPAND DATA ARRAY TO ENCOMPASS NEW POINTS
      CALL EXPAND(KEL,NUM,NDATA,MDIV)
      INC2=KEL
      INC1=0
C PUT NEW DATA INTO DATA ARRAY
      DO 11 I=1,MDIV
      NDATA(INC2)=LTEMP(I)
11    INC2=INC2+1
18    CALL REMOVE(2)
      RETURN
      END
C
C
C
```

```
      SUBROUTINE DIVNUM(NDATA,KEL,MDIV,LTEMP)
C
C DIVNUM ASKS USER WHAT HE WISHES TO DIVIDE BY
C
C INPUT...
C      NDATA=ARRAY OF CURRENT DATA VALUES
C      KEL=LOCATION OF NUMBER TO BE DIVIDED
C      LTEMP=VALUES TO BE SUBSTITUTED FOR NUMBER DIVIDED
C OUTPUT...
C      MDIV=DIVISOR OF LARGE NUMBER
C
      DIMENSION NDATA(200),LTEMP(10)
13    NTOTAL=0
C GET WHAT USER WANTS TO DIVIDE BY
      WRITE(1,1)
1     FORMAT(/,1X,*WHAT DO YOU INTEND TO DIVIDE BY (8 MAX)≤*)
10    READ (2,20) MDIV
20    FORMAT(I)
      IF (MDIV.EQ.0) GOTO 6
      IF(MDIV.GT.8.OR.MDIV.LT.0.OR.MDIV.EQ.1)  GOTO 10
C DETERMINE QUOTIENT OF DESIRED NO. DIVIDED BY NO. NO. USER DESIRES
      NQUO=NDATA(KEL)/MDIV
C STORE REPLACEMENT VALUES
      DO 3 I=1,MDIV
      NTOTAL=NTOTAL +NQUO
3     LTEMP(I)=NQUO
C ASK USER IF VALUES ARE ACCEPTABLE
      WRITE(1,4)NQUO,NTOTAL
4     FORMAT(/,1X,*QUOTIENT=*,I5,/,1X,*ORIGINAL VALUE SHOULD BE=*,I5,/,1
     10X,*IS THIS CORRECT (Y OR N)≤*)
12    READ(2,5) KANS
5     FORMAT(A1)
      IF(KANS.NE.1HN.AND.KANS.NE.1HY) GOTO 12
      IF(KANS.EQ.1HY) GOTO 6
C IF VALUES NOT ACCEPTABLE ASK FOR VALUES
      NTOTAL=0
      DO 8 K=1,MDIV
      WRITE (1,7)MDIV
7     FORMAT (/,1X,*WHAT*,I2,* VALUES DO YOU WISH TO USE:*)
      READ(2,2) LTEMP(K)
2     FORMAT (I)
8     NTOTAL=NTOTAL+LTEMP(K)
C DISPLAY ON SCREEN WHAT VALUES USER ENTERED
14    WRITE(1,9) MDIV,(LTEMP(J),J=1,MDIV),NTOTAL
9     FORMAT(/,NI5,/,*ORIGINAL VALUE SHOULD BE =*,I5,/,*ARE THESE THE VA
     1LUES YOU WISH TO USE (Y OR N)≤*)
      READ (2,11) LANS
11    FORMAT(A1)
C ERROR CHECK
      IF (LANS.NE.1HY.AND.LANS.NE.1HN) GOTO 14
      IF (LANS.EQ.1HN) GOTO 13
6     RETURN
      END
      SUBROUTINE DETECT (KEL)
C
C DETECT FINDS LOCATION OF DAA PT FROM LIGHT PEN HIT
C
C OUTPUT...
C      KEL=LOCATION OF DATA POINT USER HIT WITH PEN
C
      COMMON/FOREGND/JCB(125),JSB(1500),LITE(20)
      CALL TRACK (LITE,20)
      CALL ONTRACK(20)
1     CALL HIT(IX,IY)
C X AND Y COORDINATES DETERMINED FROM LIGHT PEN HIT
      IF((IX-36).NE.0) GOTO 4
      NX=1  $  GOTO 2
4     NX=(((IX-36)/10)/5)+1
2     NY=((777-IY)/20)/2
C DETERMINE LOCATION IN DATA ARRAY FROM X-Y COORDINATES
      KEL=(NY*10)+NX
      WRITE (1,3)KEL
3     FORMAT(/,1X,*ARRAY ELEMENT=*,I5,/,5X,*IS THIS CORRECT (Y OR N)≤*)
5     READ(2,6)NANS
6     FORMAT(A1)
      CALL REMOVE(1)
      IF (NANS.NE.1HY.AND.NANS.NE.1HN) GOTO 5
      IF (NANS.EQ.1HY) GOTO 8
C GET ANOTHER PEN IF ELEMENT UNACCEPTABLE
      WRITE (1,7)
7     FORMAT(/,2X,*DETECT AGAIN*)
      GOTO 1
8     CALL OFFTRACK
      RETURN
      END
C
C
C
      SUBROUTINE EXPAND(KEL,NUM,NDATA,MDIV)
C
C EXPAND ENLARGES DATA ARRAY TO HOLD NEW POINTS
C
C INPUT...
C      KEL=LOCATION OF ELEMENT DIVIDED
C      NUM=NUMBER OF ELEMENTS IN NDATA
C      NDATA=CURRENT DATA VALUES ALONG WITH NALUE TO BE DIVIDED
C      MDIV=DIVIWOR OF LARGE DATA VALUE
C OUTPUT...
C      NDATA=DATA VALUES WITH ROOM FOR LARGE MARKED VALUE TO BE DIVIDED
C      NUM=NUMBER OF ELEMENTS IN NEW NDATA
C
      DIMENSION NDATA(200),NHOLD(200)
      INC1=KEL
C SAVE ORIGINAL DATA ARRAY
      DO 26 J=1,NUM
26    NHOLD(J)=NDATA(J)
C EXPAND ARRAY POINTER NUM TO ENABLE DATA ARRAY TO HOLD NEW NUMBERS
      NUM=NUM+(MDIV-1)
C SHIFT DATA ARRAY ELEMENTS TO THE RIGHT TO ENCOMPASS NEW NUMBERS
      DO 10 I=KEL+MDIV,NUM
10    NDATA(I)=NHOLD(I-(MDIV-1))
      RETURN
      END
      SUBROUTINE ADD (NDATA,NUM)
C
C ADD COMBINES DATA POINTS WHICH ARE TOO SMALL
C
C INPUT...
C       NDATA=CURRENT DATA POINTS WITH THE POINTS USER WISHES TO ADD TOGETHER
C       NUM=NUMBER OF ELEMENTS IN NDATA WITH NUMBERS TO BE ADDED TOGETHER
C   (IN COMMON)
C       IOV=SCOPE OVERLAY NUMBER INDICATING WHICH ONE TO WRITE ON
C       MODE=INDICATES WHICH FORMATTED WRITE MODE TO USE
C       LIM1=LOWER LIMIT↓ LOCATION IN NDATA TO BEGIN WRITING DATA VALUES FROM
C       LIM2=UPPER LIMIT↓ LOCATION IN NDATA TO TERMINATE WRITING DATA VALUES
C OUTPUT...
C       NDATA=CURRENT DATA VALUES WITH MARKED POINTS ADDED TOGETHER
C       NUM=NEW NUMBER OF ELEMENTS IN NDATA
C
      COMMON/FOREGND/JCB(125),JSB(1500),LITE(20)
      COMMON/LIM/LIM1,LIM2
      COMMON/KMODAL/IOV,MODE
      DIMENSION NDATA(200),LSAVE(10)
C FIND NUMBER OF POINTS USER WISHES TO ADD TOGETHER
1     CALL ADDNUM(NADD)
C ZERO NO. INDICATES NEED TO LEAVE ROUTINE
      IF (NADD .EQ. 0) GOTO 21
C DETERMINE SUM OF THE COMBINED NUMBERS
      CALL SUM(NDATA,KEL,LSAVE,KANS,NSUM,NADD)
      IF (KANS.EQ.1HN) GOTO 1
C ERROR CHECK THAT NUMBERS ARE IN CONSECUTIVE ORDER
      DO 10 KK=1,NADD-1
      IF (LSAVE(KK).LT.LSAVE(KK+1)) GOTO 10
      WRITE (1,9)  $  CALL DELAY(3)  $  GOTO 1
9     FORMAT(/,*NUMBERS NOT DETECTED IN LOCATIONALLY,*,/,5X,*CONSECUTIVE
     1, ASCENDING ORDER*)
10    CONTINUE
C CONDENSE DATA ARRAY
      CALL CONDENS(NADD,LSAVE,NDATA,NUM,KEL)
      NDATA(KEL-(NADD-1))=NSUM
21    CALL REMOVE(2)
      RETURN
      END
C
C
C
      SUBROUTINE ADDNUM(NADD)
C
C ADDNUM FINDS NUMBER OF POINTS USER WISHES TO ADD TOGETHER
C
C OUTPUT...
C       NADD=NUMBER OF POINTS TO BE ADDED TOGETHER
C
1     WRITE(1,2)
2     FORMAT(/,*HOW MANY NUMBERS DO YOU INTEND TO ADD*,/,5X,*TOGETHER (1
     10 NUMBERS MAX)≤*)
      READ (2,3)NADD
3     FORMAT(I)
      IF (NADD.EQ.1.OR.NADD.LT.0) GOTO 1
      IF(NADD.GT.10)NADD=10
      RETURN
      END
C
C
```

```
      SUBROUTINE SUM(NDATA,KEL,LSAVE,KANS,NSUM,NADD)
C
C SUM ADDS BAD DATA POINTS
C
C INPUT...
C     NDATA=ARRAY OF CURRENT DATA VALUES
C     NADD=NUMBER OF DATA VALUES TO BE ADDED TOGETHER
C OUTPUT...
C     KEL=LOCATION OF LAST DATA ELEMENT DETECTED TO BE AN ADDEND
C     LSAVE=ARRAY WITH LOCATIONS OF POINTS DETECTED BY USER
C     KANS=FLAG INDICATING NUMBERS DETECTED ARE CORRECT(Y OR N)
C     NSUM=SUM OF DETECTED VALUES ADDED TOGETHER
C
      COMMON/FOREGND/JCB(125),JSB(1500),LITE(20)
      DIMENSION NDATA(200),LSAVE(10),MTEMP(10)
      NSUM=0
      NADDEND=NADD
C FIND LOCATION OF EACH BAD DATA PT TO BE ADDED
1     DO 4 K=1,NADD
      WRITE(1,10) NADDEND
10    FORMAT(/,1X,*DETECT*,1X,I2,1X,*NUMBERS*)
      NADDEND=NADDEND-1
      CALL DETECT(KEL)
C SAVE LOCATIONS OF BAD PT
      LSAVE(K)=KEL
C SAVE VALUE OF BAD PT
      MTEMP(K)=NDATA(KEL)
C SUM BAD DATA VALUES
4     NSUM=NDATA(KEL)+NSUM
C TELL USER RESULTS OF SUMMING
      WRITE(1,6)NADD,(MTEMP(K),K=1,NADD),NSUM
6     FORMAT(/,*NUMBERS≤*,NI5,/,5X,*WILL BE ADDED TOGETHER TO GIVE A SUM
     1=*,I5,/,*IS THIS CORRECT(Y OR N)≤*)
9     READ(2,8) KANS
8     FORMAT(A1)
      IF(KANS.NE.1HN.AND.KANS.NE.1HY) GOTO 9
      RETURN
      END
      SUBROUTINE CONDENS(NADD, LSAVE, NDATA, NUM, KEL)
C
C CONDENS CONDENSES DATA ARRAY TO CONFORM WITH COMBINING SEVERAL POINTS
C
C INPUT...
C     NADD=NUMBER OF DATA VALUES SUMMED TOGETHER
C     LSAVE=LOCATTON=LOCATIONS OF DATA VALUES SUMMED TOGETHER
C     NDATA=CURRENT DATA VALUES
C     NUM=NUMBER OF ELEMENTS IN NDATA
C     KEL=LAST ELEMENT DETECTED BY USER
C OUTPUT...
C     NDATA=CURRENT DATA VALUES WITH POINTS ADDED
C     NUM=NUMBER OF ELEMENTS IN NDATA
C
      DIMENSION LSAVE(10),NDATA(200)
      INC=KEL
C CONDENSE DATA ARRAY
      DO 9 J=KEL-(NADD-2),NUM
      NDATA(J)=NDATA(INC+1)
9     INC=INC+1
C CONDENSE NUMBER OF DATA POINTS POINTER
      NUM=NUM-(NADD-1)
      RETURN
      END
      SUBROUTINE FORBACK(KNUM,NLOW,IPT1,IPT2,KVAL,NUM,NDATA,KBUF,NBUF,NT
     1EMP,NVAL)
C
C FORBACK RESETS DATA ARRAYS AFTER HAVING CALLED BACK ROUTINE
C
C INPUT...
C     KVAL=NO. OF DATA VALUES RETURNED TO NBUF FROM EXPANDED NDATA
C     KNUM=NO. OF VALUES TO BE REPLACED INTO KBUF FROM EXPANDED NDATA
C     NLOW=LOCATION OF VALUE BEFORE 1ST NBUF VALUE TO REPLACED FROM NDATA
C     NVAL=TOTAL NO. OF PREVIOUS DATA VALUES TO BE RETURNED FROM EXPANDED NDATA
C     IPT1=NUM MODULO 10(NUMBER OF ELEMENTS IN NTEMP)
C     IPT2=NUMBER OF ELEMENTS ( IN UNITS OF 10) IN NBUF
C     NUM=NUMBER OF ELEMENTS IN EXPANDED NDATA
C     NDATA=ARRAY OF CURRENT DATA VALUES PLUS SOME PREVIOUS VALUES
C     NBUF=BUFFER OF CURRENT CORRECTED DATA POINTS (UNITS OF 10)
C     NTEMP=ARRAY OF TRAILING END OF VALUES (NUM MODULO 10)
C     KBUF=INTERMEDIATE BUFFER (100 VALUES) TAKEN IMMEDIATELY FROM NBUF WHEN IT
C          IS FULL
C OUTPUT...
C     NDATA=DATA VALUES IN THEIR PRE-BACK FORM
C     NUM=NUMBER OF ELEMENTS IN NDATA
C
      DIMENSION KBUF(100),NBUF(100),NTEMP(10),NDATA(200)
      IF(KNUM.EQ.0) GOTO 3
C RESET KDATA ARRAY
1     DO 2 I=1,KNUM
2     KBUF((100-KNUM)+I)=NDATA(I)
C CHECK TO SEE IF NBUF IS EMPTY
3     IF(KVAL.LE.0) GOTO 5
C RESET NBUF DATA ARRAY
      DO 4 J=1,KVAL
4     NBUF(J+NLOW)=NDATA(KNUM+J)
C CHECK TO SEE IF NTEMP IS EMPTY
5     IF(IPT1.LE.0) GOTO 7
C RESET NTEMP ARRAY
      DO 6 K=1,IPT1
6     NTEMP(K)=NDATA(KNUM+KVAL+K)
C RESET NDATA ARRAY
7     DO 8 M=1,NUM-NVAL
8     NDATA(M)=NDATA(NVAL+M)
      NUM=NUM-NVAL
      RETURN
      END
      SUBROUTINE BACK(NDATA,NUM,IPT1,IPT2,NBUF,KBUF,NTEMP,KNUM,NLOW,NVAL
     1,KVAL,NFROM,NBK)
C
C BACK COMBINES PREVIOUS LINES OF DATA WITH CURRENT DATA
C
C INPUT...
C       NDATA=CURRENT DATA VALUES BEING EXAMINED
C       NUM=NUMBER OF ELEMENTS IN NDATA
C       IPT1=NUM MODULO 10 (NUMBER OF ELEMENTS IN NTEMP)
C       IPT2=NUMBER OF ELEMENTS (IN UNITS OF 10) IN NBUF
C       NBUF=BUFFER OF CURRENT CORRECTED DATA POINTS (UNITS OF 10)
C       KBUF=INTERMEDIATE BUFFER (100 VALUES) TAKEN IMMEDIATELY FROM NBUF WHEN IT
C            IS FULL
C       NTEMP=ARRAY OF TRAILING END OF VALUES (NUM MODULO 10)
C       NBK=NUMBER OF ELEMENTS WHICH HAVE BEEN PROCESSED PREVIOUSLY
C OUTPUT...
C       NDATA=CURRENT DATA VALUES PLUS LINES OF PREVIOUS DATA
C       NUM=NUMBER OF ELEMENTS IN EXPANDED NDATA
C       KNUM=NUMBER OF VALUES TO BE OBTAINED FROM KBUF FOR EXPANDED NDATA
C       NLOW=LOCATION OF VALUE BEFORE 1ST NBUF VALUE TO BE PLACED IN NDATA
C       NVAL=TOTAL NUMBER OF PREVIOUS DATA VALUES TO BE ADDED TO NDATA
C       KVAL=NUMBER OF DATA VALUES TO BE OBTAINED FROM NBUF FOR EXPANDED NDATA
C       NFROM=FLAG INDICATING BACK ROUTINE HAS BEEN PROCESSED
C
      DIMENSION NDATA(200),NBUF(100),KBUF(100),NTEMP(10), KDATA(200)
      COMMON/LIM/LIM1,LIM2
      NFROM=1HB
C ASK USER HOW MANY LINES BACK HE WISHES TO SEE
      CALL LINE(NFROM,NLINES,NVAL)
      IF (NBK.LT.NLINES) NVAL=NBK*10
      LIMIT=NVAL+NUM
C MAXIMUM NO. OF 15 LINES OF DATA
      IF(LIMIT.GT.150)NVAL=(150-NUM)-MOD((150-NUM),10)
C TEMPORARIRY STORE DATA ARRAY
      DO 3 K=1,NUM
3     KDATA(NVAL+K)=NDATA(K)
      IF(NVAL.GT.IPT2) GOTO 4
      KNUM=0  $  KVAL=NVAL-IPT1
      NLOW=IPT2-KVAL
      IF(IPT1.GT.0) GOTO 6
      GOTO 8
4     KNUM=NVAL-(IPT2+IPT1)
C RETREIVE LINES NEEDED FROM KBUF(INTERMEDIATE DATA ARRAY)
      DO 5 I=1,KNUM
5     KDATA(I)=KBUF((100-KNUM)+I)
      NLOW=0  $  KVAL=IPT2
      IF(IPT1.LE.0) GOTO 8
C RETRIEVE MODULO 10 NUMBERS BDING TEMPORARILY STORED
6     DO 7 J=1,IPT1
7     KDATA(KNUM+KVAL+J)=NTEMP(J)
8     IF(KVAL.LE.0) GOTO 10
C RETRIEVE LINES NEEDED FROM NBUF(STORAGE ARRAY)
      DO 9 K=1,KVAL
9     KDATA(KNUM+K)=NBUF(NLOW+K)
10    NUM=NVAL+NUM
C RETURN COMPLETED DATA ARRAY WITH PREVIOUS LINES DESIRED TO NDATA
      DO 11 J=1,NUM
11    NDATA(J)=KDATA(J)
      RETURN
      END
      SUBROUTINE FORWARD(NDATA,NUM,IPT1,IPT2,NBUF,KBUF,NTEMP,NSTOP,KSTOP
     1,MBUF)
C
C FORWARD STORES AND OUTPUTS CORRECTED DATA
C
C INPUT...
C     NDATA=CURRENT DATA VALUES
C     NUM=NUMBER OF ELEMNTS IN NDATA
C     IPT1=NUM MODULO 10 (NUMBER OF ELEMENTS IN NTEMP)
C     IPT2=NUMBER OF ELEMENTS (IN UNITS OF 10) IN NBUF
C     NBUF=BUFFER OF CURRENT CORRECTED DATA POINTS (UNITS OF 10)
C     KBUF=INTERMEDIATE BUFFER (100 VALUES) TAKEN IMMEDIATELY FROM NBUF WHEN IT
C          IS FULL
C     NTEMP=ARRAY OF TRAILING END OF VALUES (NUM MODULO 10)
C     NSTOP=FLAG INDICATING TO PRINT OUT KBUF, NBUF, NTEMP (END OF FILE PROCESS)
C     KSTOP=FLAG INDICATING TO PRINT OUT KBUF,NBUF, NTEMP(ENE OF PROFP N)4
C     MBUF=MEMORY BUFFER FOR OUTPUT
C OUTPUT...
C     IPT1=NUM MODULO 10 (NUMBER OF ELEMENTS IN NTEMP)
C     IPT2=NUMBER OF ELEMENTS (IN UNITS OF 10) IN NBUF
C     NBUF=BUFFER OF CURRENT CORRECTED DATA POINTS (UNITS OF 10)
C     KBUF=INTERMEDIATE BUFFER (100 VALUES) TAKEN IMMEDIATELY FROM NBUF WHEN IT
C          IS FULL
C     NTEMP=ARRAY OF TRAILING END OF VALUES (NUM MODULO 10)
C     MBUF=MEMORY BUFFER FOR OUTPUT
C
      DIMENSION NDATA(200),NBUF(100),NTEMP(10),KBUF(100),MBUF(100)
      INCR=1
      IF(IPT1.GT.0) GOTO 4
      NUMBR=0
      IF (IPT2.GE.100) GOTO 8
      GOTO 9
C PUT NUMBERS MOD 10 TEMPORARILY STORED FROM LAST TIME INTO BUFFER FOR STORAGE
4     CALL OLDTEMP(NUMBR,NTEMP,NDATA,NBUF,IPT1,IPT2)
      IF(IPT2.LT.100) GOTO 9
C CORBUF WRITES CORRECTED DATA ON A NEW TAPE
8     CALL CORBUF (NBUF,KBUF,MBUF)
      IPT2=0
9     IHOLD=NUM-NUMBR
      ITEMP1=MOD(IHOLD,10)
      ITEMP2=IHOLD-ITEMP1
10    IF(ITEMP2.EQ.IHOLD) GOTO 12
      DO 11 J=1,ITEMP1
11    NTEMP(J)=NDATA((NUM-ITEMP1)+J)
C PUT CORRECTED NDATA ONTO NBUF ARRAY
12    DO 13 K=INCR,ITEMP2
      NBUF(IPT2+K)=NDATA(NUMBR+K)
      IF((IPT2+K).EQ.100) GOTO 15
13    CONTINUE
      IPT2=IPT2+ITEMP2
14    IPT1=ITEMP1
C INITIALIZE DATA ARRAY
      DO 17 I=1,NUM
17    NDATA(I)=0
C WRITE OUT INTERMEDIATE BUFFERS WHEN PROGRAM STOPS OR WHEN GETTING NEW FILE
      IF (NSTOP.NE.1HY.AND.KSTOP.NE.1HY) GOTO 18
      WRITE(4,16)(MBUF(K),K=1,100)
      WRITE(4,16)(KBUF(K),K=1,100)
      IF (IPT2.GT.0) WRITE(4,16)(NBUF(I),I=1,IPT2)
      IF (IPT1.GT.0) WRITE(4,16)(NTEMP(I),I=1,IPT1)
16    FORMAT (10I5)
```

```
18     RETURN
15     INCR=K+1
C CORBUF WRITES CORRECTED DATA ON A NEW TAPE
5      CALL CORBUF (NBUF,KBUF,MBUF)
6      IPT2=0
       IF((INCR-1).EQ.ITEMP2) GOTO 14
       IPT2=-(INCR-1)
       GOTO 12
       END
C
C
C
       SUBROUTINE OLDTEMP(NUMBR,NTEMP,NDATA,NBUF,IPT1,IPT2)
C
C OLDTEMP STORES PREVIOUS MO 10 NO. INTO STORAGE BUFFER
C
C      INPUT...
C      NTEMP=ARRAY OF TRAILING END OF VALUES (NUM MODULO 10)
C      NDATA=CURRENT DATA VALUES
C      IPT1=NUM MODULO 10 (NUMBER OF ELEMENTS IN NTEMP)
C      IPT2=NUMBER OF ELEMENTS (IN UNITS OF 10) IN NBUF
C OUTPUT...
C      NUMBR=NUMBER OF ELEMENTS NEEDE FROM NDATA TO MAKE TEN ELEMENTS IN NTEMP
C      NBUF=PREVIOUS DATA VALUES PLUS 10 NEW VALUES FROM NTEMP
C      IPT2=INCREMENTED BY 10 (10 NEW VALUES ADDED TO NBUF)
C
       DIMENSION NDATA(200),NTEMP(10),NBUF(100)
CNUMBR =NO. OF ELEMENTS FROM NEW DATA ARRAY OBTAIN A LINE OF 10 ELEMENTS
       NUMBR=10-IPT1
C OBTAIN NEEDED NO. OF ELEMENTS FROM NDATA IN NTEMP ARRAY
       DO 2 K=1,NUMBR
2      NTEMP(IPT1+K)=NDATA(K)
C STORE NEW LINE OF NUMBERS INTO NBUF
       DO 3 J=1,10
3      NBUF(IPT2+J)=NTEMP(J)
       IPT2=IPT2+10
       RETURN
       END
```

```
      SUBROUTINE CORBUF (NBUF,KBUF,MBUF)
C
C CORBUF WRITES CORRECTED DATA BUFFER ON A NEW TAPE
C
C INPUT...
C      NBUF=BUFFER OF 100 CURRENT AND OLD DATA POINTS TO BE PUT INTO KBUF
C      KBUF=INTERMEDIATE DATA BUFFER (USED FOR BACK ROUTINE) TO GO INTO MBUF
C      MBUF=MEMORY DATA, READY FOR OUTPUT
C OUTPUT...
C      KBUF=FORMER NBUF VALUES
C      MBUF=FORMER KBUF VALUES
C
      DIMENSION NBUF(100),KBUF(100),MBUF(100)
21    IF(MBUF(1).EQ.0) GOTO 17
C STORE CORRECTED DATA
      WRITE(4,16)(MBUF(K),K=1,100)
16    FORMAT(10I5)
C MBUF IS MEMORY BUFFER↓IT CANNOT BE ACCESSED BY USER
17    DO 18 J=1,100
      MBUF(J)=KBUF(J)
18    KBUF(J)=NBUF(J)
22    RETURN
      END
C
C
C
      SUBROUTINE QUIT(MANS,NFIL)
C
C QUIT ASKS USER WHETHER TO STOP PROGRAM OR GET NEW FILE
C
C INPUT...
C      NFIL=FLAG INDICATING USER WANTS TO STOP(=S) OR THE NEXT FILE(=F)
C OUTPUT...
C      MANS=FLAG INDICATING YES(=Y) OR NO(=N)
C
      IF(NFIL.EQ.1HS) GOTO 2
C ASK USER IF HE WANTS ANOTHER FILE
      WRITE (1,1) $  GOTO 10
1     FORMAT(/,5X,*ARE YOU SURE YOU WANT THE NEXT FILE(Y OR N)≤*)
C ASK USER IF HE WANTS TO STOP
2     WRITE (1,4)
4     FORMAT(/,5X,*ARE YOU SURE YOU WANT TO STOP (Y OR N)≤*)
10    READ(2,3)MANS
3     FORMAT(A1)
C ERROR CHECK
      IF(MANS.NE.1HY.AND.MANS.NE.1HN) GOTO 10
      RETURN
      END
```

Appendix B

Input/Output Data Example

Input Data...

752	734	720	706	689	662	676	658	654	656
674	668	655	645	636	683	703	705	462	[illegible]
717	743	735	699	707	706	715	721	719	726
731	726	705	678	670	657	631	652	686	702
738	750	752	823	696	784	785	749	760	741
738	761	746	755	756	754	697	727	721	737
722	696	685	733	730	778	771	816	833	754
[illegible]	1114	729	732	740	727	741	707	741	702
672	654	694	689	699	691	688	684	690	711
679	703	695	692	659	670	677	655	663	679
695	709	689	687	686	702	726	696	691	682
669	679	689	685	671	1331	667	667	1328	654
675	682	734	704	700	693	747	748	744	727
724	740	739	728	739	727	745	733	718	736
726	729	724	699	706	699	708	736	755	780
776	803	785	700	693	753	842	826	785	748

Corrected Output Data...

710	752	734	720	706	689	662	676	658	654
656	674	668	655	645	636	683	703	705	727
[illegible]	717	743	735	699	707	706	715	721	719
726	731	726	705	678	670	657	631	652	686
702	738	750	752	823	696	784	785	749	760
741	738	761	746	755	756	754	697	727	721
737	722	696	685	733	730	778	771	816	833
754	[illegible]	[illegible]	729	732	740	727	741	707	741
702	672	654	694	689	699	691	688	684	690
711	679	703	695	692	659	670	677	655	663
679	695	709	689	687	686	702	726	696	691
682	669	679	689	685	671	[illegible]	[illegible]	667	667
[illegible]	[illegible]	654	675	682	734	704	700	693	747
748	744	727	724	740	739	728	739	727	745
733	718	736	726	729	724	699	706	699	708
736	755	780	776	803	785	700	693	753	842

References

1. Control Data Cyber 70 Computer Systems Models 72, 73, 74/ 6000 Computer Systems FORTRAN Reference Manual, Publication No. 6017900, Revision F, 21 July 1972.

2. Laurens, Judy, ed. User's Manual, University of Texas at Austin Computation Center, Revised December 1974.

3. Marchak, James. FORTRAN Reference Manual for the MNF and RUN Compilers at the University of Texas at Austin, University of Texas at Austin Computation Center, Fall, 1976.

4. (SDS) 930 Arachnid Graphics Package, University of Texas at Austin Computation Center, 6 May 1976.

5. University of Texas FORTRAN IV for the SDS 930 Spider System, University of Texas at Austin Computation Center, 6 May 1976.